Escuela Politécnica Superior

TRABAJO FIN DE GRADO

Algoritmo de Monitorización de Conductor en Sistema Android

Autor: Sergio Romero Salas

Director: Fernando García Fernández

Madrid, Septiembre 2015

Departamento de Ingeniería Electrónica y Automática

grado en ingeniería en tecnologías industriales

Algoritmo de Monitorización de Conductor en Sistema Android

Autor: Sergio Romero Salas

Director: Fernando García Fernández

Madrid, Septiembre 2015

Título: Algoritmo de Monitorización de Conductor en Sistema Android
Autor: Sergio Romero Salas
Director: Fernando García Fernández

EL TRIBUNAL

Presidente:

Vocal:

Secretario:

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ..... de ..... de ... en ....., en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

"Let the future tell the truth and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I really worked, is mine."

Nikola Tesla

Agradecimientos

En primer lugar quisiera agradecer el apoyo y la comprensión que me han aportado mi familia, pareja, compañeros y amigos durante estos cuatro duros aunque satisfactorios años.

Mi familia siempre ha estado apoyándome en lo que fuera necesario y me ha permitido alcanzar este sueño que comenzó en septiembre de 2011, sin importarles los días sin pasar por casa, las noches sin dormir o los momentos más difíciles donde era de todo menos amable. Siempre os estaré agradecido.

Gracias también a Silvia, mi otro gran punto de apoyo. Gracias por aguantar mis malos momentos, por su compresión con mi dedicación a los estudios y por su cariño tan importante para mí. Has estado a mi lado en las buenas y en las malas, siempre creyendo en mí y enseñándome que nunca hay que desistir en lo que uno se propone. Muchas gracias por todo, te quiero.

Tengo muy claro que no habría llegado hasta aquí si no fuera por mis compañeros de carrera, sin su ayuda esto no hubiera sido posible. Me quedará para el recuerdo todas las tardes en la biblioteca en segundo y tercer curso, todas las risas, todas las noches sin dormir juntos en la distancia y todos los buenos momentos que son, sin duda, mejores de lo que podría imaginar. Sois lo mejor que me llevo de estos cuatro años.

No puedo olvidarme tampoco de mis amigos desde pequeño, de todas las noches en Getafe con conversaciones y circunstancias que nos hicieron madurar, de todas las historias que hemos vivido, de los buenos y malos momentos y de todo lo que nos queda por vivir.

En último lugar quisiera agradecer a Fernando García la oportunidad que me ha ofrecido de realizar este trabajo fin de grado bajo su tutela y ayuda y su confianza depositada en mí durante estos meses, así como al departamento de sistemas inteligentes la Universidad Carlos III de Madrid por ofrecerme el soporte necesario.

Todas estas personas me han hecho ser lo que soy hoy, y espero que sigan aportándome todo esto durante mucho tiempo más.

Resumen

En las siguientes páginas se expone el trabajo realizado y las diferentes soluciones aportadas para el desarrollo de una funcionalidad orientada a la detección de la fatiga en un conductor de un vehículo.

Debido al aumento de las muertes en accidentes de tráfico, nuestra sociedad se ve obligada a buscar una solución para dicho problema. La globalización y el enorme crecimiento del sector de comunicaciones han hecho que un gran número de la población posea un dispositivo móvil, hecho que se puede aprovechar como herramienta para lograr el objetivo de reducir las muertes en accidentes de tráfico y aumentar la seguridad vial.

Para dicho fin se desarrollará una funcionalidad nueva orientada a la detección de la fatiga en el conductor en una aplicación ya previamente implementada por el Laboratorio de Sistemas Inteligentes de la Universidad Carlos III de Madrid. El sistema operativo en el que se desarrolla esta aplicación es el sistema operativo Android, presente en muchos dispositivos como móviles, tabletas, relojes, etc. En cuanto al lenguaje en el que se implementará el código necesario es el lenguaje Java y el entorno de programación será el programa Android Studio. Además, para el trabajo de visión por computador utilizaremos la librería OpenCV, que nos ofrece funciones destinadas al tratamiento y análisis de imágenes con el objetivo de obtener información de ellas.

Palabras clave: seguridad vial, fatiga, Android, Java, OpenCV, lenguaje de programación, Android Studio, sistema operativo, librería de funciones.

Abstract

In the next pages it is described the research made and the different solutions provided with the aim of the implementation of a functionality guided to the detection of fatigue in a vehicle driver.

Due to the increase of the deaths because road accident, there is the need in our society to find a solution for the problem of the road safety. Globalization and the huge growth of the communication industry have created a situation where the majority of the population have an own mobile phone, fact that can be useful for reach the objective of reducing the deaths in the road and to increase the road security.

In order to reach this goal, it will be development a new functionality oriented to the detection of the fatigue in the driver into an already implemented application by the Intelligent Systems Laboratory of Universidad Carlos III de Madrid. The operative system in which the application is developed is the Android operative system, present in a lot of devices like smartphones, tablets, smartwatches, etc. The programming language in which the code of the application will be written is the language Java and the programming environment will be Android Studio. In addition, for the computer vision work we will use the OpenCV library, which offers us functions oriented to the images manipulation and analysis with the aim of get information about them.

Keywords: road safety, fatigue, Android, Java, OpenCV, programming language, Android Studio, operative system, functions library, programming environment.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

En la actualidad los accidentes de tráfico suponen una de las principales causas de muerte en el mundo, siendo la novena causa en el año 2012 por debajo de enfermedades como el cáncer o las enfermedades del corazón (Figura 1.1). En 2012 se produjeron en el mundo 1,3 millones de defunciones en accidentes de tráfico. Esta cifra se ha incrementado desde el año 2000, año en el cual hubo 1 millón de defunciones por accidente de tráfico[15]. Este número se ha elevado, entre otros motivos, por el crecimiento del sector del automóvil, por el aumento de la globalización y por el crecimiento de la economía.

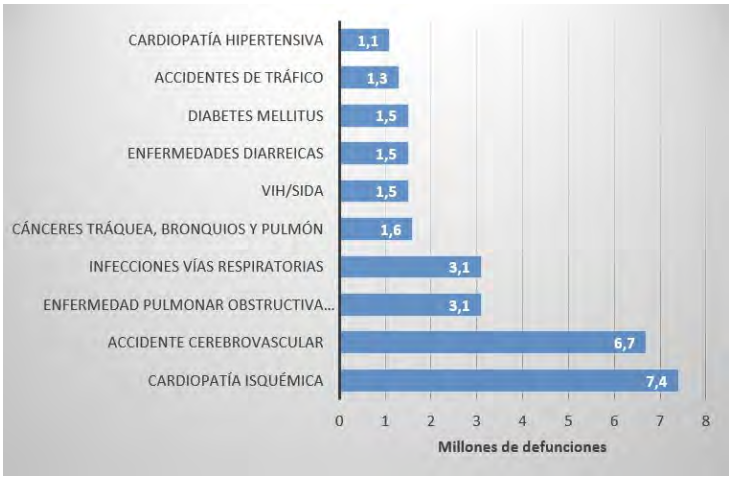

Figura 1.1: 10 causas principales de defunción en el mundo en 2012[15].

En España, perder la vida en un accidente de tráfico supone la quinta

causa de muerte no natural en el año 2013, según el INE. Durante el 2013 hubo en España 1.807 muertos en accidentes de tráfico, aproximadamente un muerto cada 5 horas[21]. Aunque este número se ha reducido (en 2007 ocupaba el primer puesto) debido a las fuertes campañas de la dirección general de tráfico, sigue siendo un número demasiado elevado.

Dentro de las muertes por accidente de tráfico, una de los principales motivos es el cansancio o la fatiga del conductor del vehículo (en España alrededor del 20 % en los últimos años). La conducción durante grandes distancias sin descanso a veces debido a trabajo o en otras ocasiones al ocio, hace aumentar el riesgo de ocasionar un accidente causado por la fatiga del conductor. Por lo tanto, es crucial concienciar al conductor de que es una importante causa de riesgo y darle todas las herramientas para que pueda controlarlo.

Teniendo en cuenta estos datos, parece claro que es un objetivo de vital importancia intentar reducir este número de víctimas en accidentes de tráfico. Esta es la motivación que mueve este proyecto.

Por otra parte, la universidad Carlos III de Madrid (UC3M) tiene una gran vía de investigación aplicada a la seguridad vial, con muchos proyectos y una gran cantidad de capital y esfuerzo invertido. En concreto, el laboratorio de sistemas inteligentes (LSI) tiene muchos proyectos dedicados a la seguridad vial, como pueden ser sus coches IVVI 2.0 o el iCab.

En este marco de trabajo se desarrolló una aplicación de seguridad vial con algunas funcionalidades como la detección de coches, detección de peatones o el tratamiento de imágenes. Esta aplicación de implementó con el objetivo de ser una base para añadir en un futuro más funcionalidades para la seguridad vial, como la que se va a implementar en este trabajo.

Aunque en la actualidad varios vehículos cuenten con esta funcionalidad incorporada, es interesante desarrollar una aplicación para dispositivo móvil y así poder hacer accesible esta ayuda a la conducción al mayor número de conductores posibles, también a aquellos que cuenten con un automóvil más antiguo.

El objetivo por tanto de este proyecto es aportar soluciones al problema de la fatiga al volante, desarrollando una funcionalidad en la aplicación base previamente implementada para que un dispositivo móvil (Smartphone) pueda detectar la presencia de fatiga en el conductor y pueda avisarle y concienciarle sobre el peligro que esto ocasiona.

1.2. Objetivos del proyecto

Los principales objetivos de este proyecto son los siguientes:

- Comprender las técnicas de visión por computador que ofrece la librería de funciones OpernCV y aplicarlas a la detección de fatiga.
- Estudiar el lenguaje de programación Java y utilizarlo para desarrollar una aplicación de seguridad vial.
- Desarrollar una aplicación en sistema operativo Android, utilizando la herramienta para desarrollo Android Studio y los conceptos de visión por computador, así como el lenguaje de programación Java..

1.3. Estado del arte

Como ya hemos comentado anteriormente, hoy en día es un aspecto muy importante para la sociedad mejorar la seguridad vial y reducir el número de muertos en las carreteras. Para ello se implementan cada día nuevos sistemas con el fin de reducir los accidentes de tráfico. En este aspecto ha sido muy importante el avance de la electrónica en los automóviles y la integración de cada vez más sistemas electrónicos en los mismos.

En este apartado expondremos los últimos avances en el desarrollo de sistemas para mejorar la seguridad vial, pasando después a explicar los últimos proyectos desarrollados para la detección de fatiga en el conductor y, por último, analizaremos las diferentes ramas de investigación que tiene la universidad Carlos III de Madrid en seguridad vial, centrándonos en el Laboratorio de Sistemas Inteligentes.

1.3.1. Sistemas para la mejora de la seguridad vial

Los fabricantes de automóviles en los últimos años han apostado por la seguridad de los conductores de sus vehículos, implementando nuevos sistemas inteligentes embebidos en los automóviles que aumenten la seguridad del cliente al ponerse al volante, aparte de los sistemas de seguridad tradicionales como el cinturón de seguridad, el airbag o el ABS. Explicaremos en las siguientes líneas algunos de estos sistemas inteligentes.

Uno de los sistemas implementado en estos últimos años ha sido el de detección de coche en ángulo muerto, conocido en inglés como BLIS (Blind Spot Information System)[3]. Este sistema fue implementado originalmente por la marca Volvo, pero se ha ido ampliando a muchas más marcas de vehículos como Mercedes-Benz, Ford, Opel, Citroën, Volkswagen,etc. Este

sistema puede estar compuesto por una cámara, un radar o por sensores de ultrasonidos. Al detectar el sistema un vehículo en el ángulo muerto, se avisa mediante un pitido o una luz al conductor para que sepa de su existencia. Un sistema similar a este mencionado es el sistema de asistente para cambio de carril, el cual funciona de forma análoga al anterior pero con un mayor campo de visión para informar al conductor cuando tenga encendido el intermitente para cambiar de carril si se acerca algún vehículo por el mismo (Fig. 1.2).

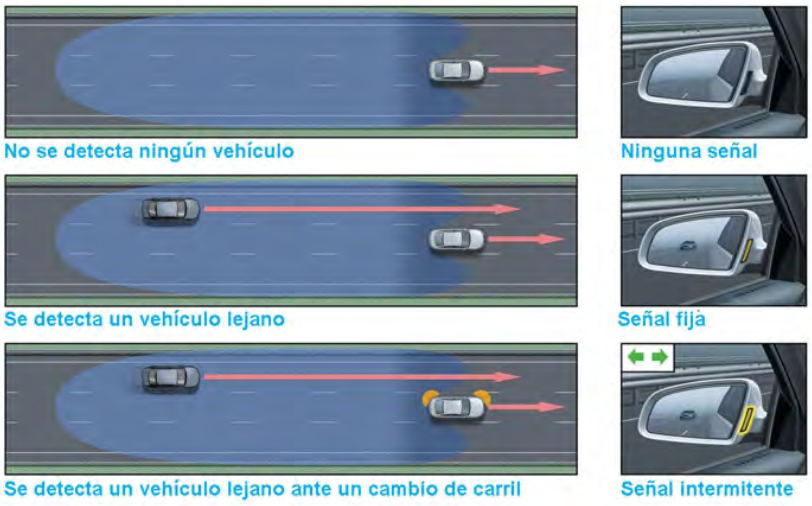

Figura 1.2: Funcionamiento del asistente de cambio de carril[3].

Un ejemplo de la integración de la visión por computador y la mecánica del coche es el sistema inteligente de limitación de velocidad. Este sistema utiliza el análisis de imágenes para analizar las señales que aparecen en la carretera mediante una cámara y avisa al conductor cuando éste excede la velocidad máxima de la vía. Este sería uno de las dos funciones de este sistema, ya que otra función puede ser limitar directamente la velocidad del vehículo para que no pueda exceder la velocidad máxima. El valor de la velocidad máxima de la vía también se puede obtener mediante una base de datos y un gps localizador del automóvil. Este sistema se encuentra actualmente en vehículos como el modelo S-Max del fabricante Ford[14], diseñado en 2015 y que incluía por primera vez esta novedoso sistema.

Continuamos exponiendo otro sistema inteligente de ayuda a la conducción como es el sistema de frenado de emergencia. Dicho sistema consta de un sensor láser, cámara o radar en la parte delantera del vehículo el cual detecta si hay algún objeto delante del mismo, como puede ser otro automóvil o un peatón, y, si el conductor no acciona el freno, el sistema lo acciona para

evitar una colisión o disminuir daños. Este sistema está presente en multitud de fabricantes de vehículos como Mazda, Volkswagen, Volvo, etc.

Figura 1.3: Sistema "Front Assist"del Volkswagen Golf de frenado de emergencia[24].

También podemos encontrar muchos otros sitemas actualmente en los vehículos como pueden ser el asistente de luz de carretera que baja la dirección de las luces de larga distancia en el caso de que se encuentre un vehículo, el asistente para mantenimiento en el carril o el eCall, dispositivo que será obligatorio en los vehículos en Europa en 2018 y que llamará al servicio de emergencias automáticamente al producirse un accidente.

Entre los desarrollos futuros, podemos encontrar un sistema en desarrollo actual de la compañía Bosch que se empezará a fabricar en 2016 conocido como el sistema "anti-kamikazes"[9]. Este sistema se aprovecha de la conexión a Internet de la que en un futuro todos los automóviles dispondrán. El sistema detecta cuando un conductor está circulando durante varios metros en sentido contrario a los demás vehículos, obteniendo estos datos por del gps del coche y compartiéndolos mediante la conexión a Internet, y posteriormente avisa al conductor distraído o "kamikaze" y al resto de conductores de la vía de esta situación.

Un proyecto más ambicioso en cuanto a seguridad vial es el que ha puesto en marcha la compañía Google[7]. Está desarrollando un vehículo que se conduce autónomamente, lo único que debe hacer el usuario es decirle la dirección a la que desea ir y el automóvil le llevará allí, reduciendo así los errores humanos y aumentando la seguridad y comodidad. Este dispositivo dispone de cámaras para controlar los obstáculos en el camino, las señales de tráfico y los demás coches (Fig. 1.4). En un futuro podría implementarse

un proyecto a gran escala en el que todos los vehículos sean conducidos autónomamente, lo cual involucraría cambiar la normativa de circulación, y se pueda así reducir en un gran número los accidentes de tráfico.

Figura 1.4: Automóvil sin conductor de Google[7].

1.3.2. Detección de la fatiga en el conductor

En cuanto a la detección de fatiga del conductor hay multitud de marcas que implementan sistemas con este fin en sus automóviles, como Volkswagen, Nissan, Skoda, etc.

Estos sistemas implementan sensores en el volante los cuales "aprenden" el comportamiento del conductor en situaciones normales y cuando esta situación no se produce (pérdida de presión al volante, correcciones bruscas de la dirección o zigzagueo por ejemplo) alerta al conductor de que tiene síntomas de fatiga con una alarma sonora (Fig 1.5).

Otro sistema para la detección de la fatiga se trata de los sistemas de detección facial mediante una cámara en el volante apuntando a la cara del conductor. Estos sistemas funcionan localizando la cara del conductor y analizando su parpadeo, para ver si se encuentra con síntomas de fatiga. Estos sistemas también son capaces de reconocer bostezos o distracción de la carretera analizando la dirección de la mirada del conductor. La marca de neumáticos Continental o otros muchos fabricantes de vehículos están integrando esta tecnología en sus modelos.

Figura 1.5: Detección de fatiga utilizando sensores en el volante[4].

1.3.3. Investigación en el LSI de la UC3M

El LSI de la UC3M fue creado en el año 2000 por profesores de la Escuela Politécnica Superior y trabaja en temas relacionados con la robótica, el control inteligente, la visión por computador, los sistemas inteligentes de transporte, etc[13]. Expondremos un breve resumen de los proyectos de investigación en seguridad vial que tiene el Laboratorio de Sistemas Inteligentes de la Universidad Carlos III de Madrid.

El LSI dispone de una rama de desarrollo de vehículos inteligentes, entre los que se encuentran el IVVI, IVVI 2.0 (segunda versión) y el iCab[12]. Los dos modelos de IVVI implementan un vehículo el cual está controlado por cámaras del espectro visible y cámaras infrarrojas, con el objetivo de detectar los posibles peligros en la carretera y avisar al conductor(Podemos observar en la Fig. 1.6 el modelo IVVI 2.0). En los vehículos IVVI han confluido muchos proyectos de seguridad vial implementados por el LSI como el de detección de peatones en baja visibilidad con cámara infrarroja, vigilancia del ángulo muerto, detección de señales de tráfico, etc. El vehículo iCab es un vehículo de Golf modificado para que pueda ser controlado mediante un ordenador y sirva de asistente de visita por el Campus de la Escuela Politécnica Superior.

Figura 1.6: IVVI 2.0 del LSI de la UC3M[12].

1.4. Estructura del documento

A continuación y para facilitar la lectura del documento, se detalla el contenido de cada capítulo.

- Para comenzar, en el capítulo primero se ha realizado una introducción al proyecto tratado, con los objetivos, motivaciones y estado del arte.
- En el segundo capítulo se hará una pequeña descripción de las herramientas utilizadas, como el entorno de programación, la librería OpenCV, sobre la cual se realizará una pequeña introducción a las herramientas que nos ofrece, o el lenguaje Java, así como el dispositivo móvil en el que probaremos nuestra aplicación.
- Posteriormente, en el tercer capítulo describiremos los procedimientos llevados a cabo para el desarrollo de la aplicación, explicando la consecución de una aplicación básica que cumpla los requisitos propuestos, la implementación de mejoras en esta aplicación y el resultado final del mismo.
- Continuando con el proyecto, realizaremos en el cuarto capítulo de este documento un análisis de los resultados obtenidos.
- También realizaremos una exposición de los costes del proyecto en el quinto capítulo.
- Para finalizar, en el sexto y último capítulo, se expondrán las conclusiones del proyecto, dejando la puerta abierta a desarrollos futuros y mejoras de la aplicación.

Al final del documento se pueden encontrar los apéndices, con el código implementado en este proyecto.

Capítulo 2

Descripción de las herramientas utilizadas

En el siguiente capítulo se describirán las herramientas y el lenguaje de programación utilizado, así como las características del dispositivo donde se han realizado las pruebas de la aplicación.

2.1. Android Studio

2.1.1. Introducción y objetivo de Android Studio

El sistema operativo elegido para desarrollar la aplicación de seguridad vial es el sistema operativo Android. Android es un sistema operativo de Google que está presente en multitud de aparatos, como Smartphones, tablets, relojes, etc, lo cual hace que sea muy interesante para el objetivo que conlleva este trabajo.

Para el desarrollo de aplicaciones en este lenguaje, Google pone a disposición del desarrollador el entorno de programación conocido como Android Studio. Android Studio gestiona de una manera eficiente todos los tipos de archivos presentes en una aplicación Android, además de ofrecernos la opción de depurar o simular la aplicación, entre muchas otras. El desarrollo de la aplicación se realizará en la versión 1.2.2 de Android Studio.[1]

Android Studio está disponible para plataformas como Windows, Linux y Mac. Su objetivo es poder facilitar la programación de aplicaciones Android para los desarrolladores expertos a la vez que permite, debido a su sencillez, desarrollar aplicaciones simples o medianamente complejas en poco tiempo para programadores inexpertos.

Con su aparición en 2013 sustituyó a Eclipse como IDE (Integrated Development Environment) oficial de desarrollo de aplicaciones Android, incorporando Gradle como nuevo compilador. Este cambio se produjo debido a la obsolescencia de Eclipse y la mayor eficiencia, rapidez y estabilidad que otorga Android Studio.

2.1.2. Gestión de archivos y carpetas

Para la gestión de los archivos de una aplicación Android Studio[27] nos provee de una pestaña a la izquierda de la ventana ("project") en la cual podemos elegir el modo en el que queremos organizar los documentos y los que queremos ocultar, dependiendo de la tarea que queramos realizar.

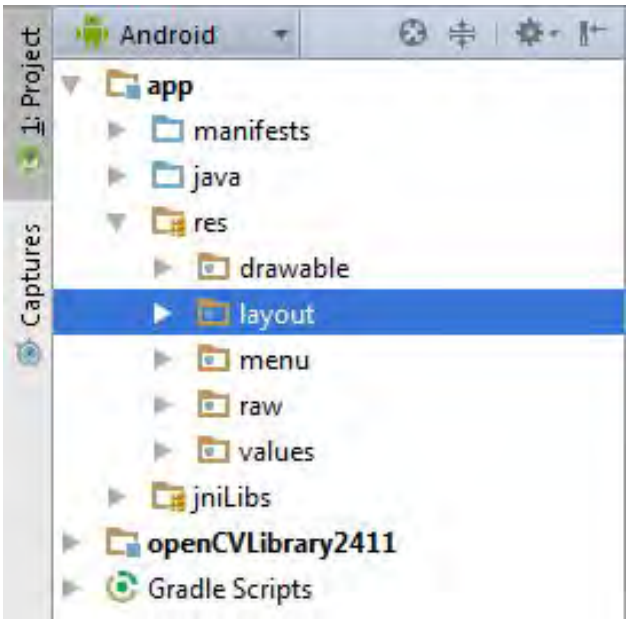

Figura 2.1: Pestaña para gestión de archivos de Android Studio.

La vista en el modo Android (Figura 2.1) es la más útil en nuestro caso, ya que nos muestra todas las carpetas y archivos útiles para nuestro desarrollo. Procedemos ahora a resumir el contenido de las mismas.

- Dentro de la carpeta app, donde se encuentran todos los archivos de la aplicación, se encuentra la carpeta manifests. En esta carpeta se puede encontrar el archivo "AndroidManifest.xml", el cual es una introducción de la aplicación, donde se definen propiedades como los permisos de la misma, la versión mínima de Android para la que es compatible y

aspectos de las actividades que forman la aplicación como su nombre u orientación.

- También podemos encontrar dentro de la carpeta app la carpeta java, en la cual podemos encontrar todas las actividades de nuestro programa, que serán los archivos donde se implementen los algoritmos necesarios para el correcto funcionamiento del programa en lenguaje Java.
- Una carpeta también importante para el desarrollo del programa es la carpeta res (resources) en la cual se encuentran todos los archivos que usaremos para la representación gráfica de la aplicación. Están clasificados de la siguiente manera:
  - En la carpeta drawable podemos encontrar tanto las imágenes que se vayan a usar en la aplicación como xml con la representación de botones o elementos de la pantalla.
  - La carpeta layout es crucial para el desarrollo de la aplicación, ya que aquí se encuentran los xml de los elementos que aparecerán en la pantalla del dispositivo como los botones, texto, contenedores, etc. Para estos layouts Android Studio nos ofrece un entorno de edición muy cómodo e intuitivo, el cual se explicará posteriormente.
  - Dentro de la carpeta menu podemos encontrar, como su propio nombre indica, los xml de los menús desplegables correspondientes a las diferentes actividades, para su previsualización y edición.
  - En la carpeta raw se guardarán archivos como audio, video o xml. La principal diferencia con otras carpetas de xml es que estos archivos no se compilarán, se incorporan al programa tal y como son.
  - Por último, en la carpeta values se recopila todo tipo de definiciones de valores constantes. Un archivo muy útil que se encuentra en esta carpeta es el archivo strings.xml, en el cual encontramos la definición de los strings que se usarán mediante su referencia para la representación de los textos presentes en los layout. Esto tiene una gran utilidad a la hora de pasar la aplicación a otro idioma, ya que únicamente tendremos que copiar este xml con los strings traducidos al idioma deseado en otra carpeta llamada values seguida de un guión y el código del idioma deseado. Así, automáticamente Android accederá a esta carpeta cuando el idioma del dispositivo sea el señalado. Además, para facilitar este trabajo de traducción, Android Studio nos provee de la herramienta llamada "Translations Editor"(Fig. 2.2) en el cual podemos traducir con mayor facilidad el contenido de los strings, añadir un nuevo idioma, marcar los strings como no traducibles (en el caso de los nombres o correos) para que no varíen su valor en cualquier idioma, etc.

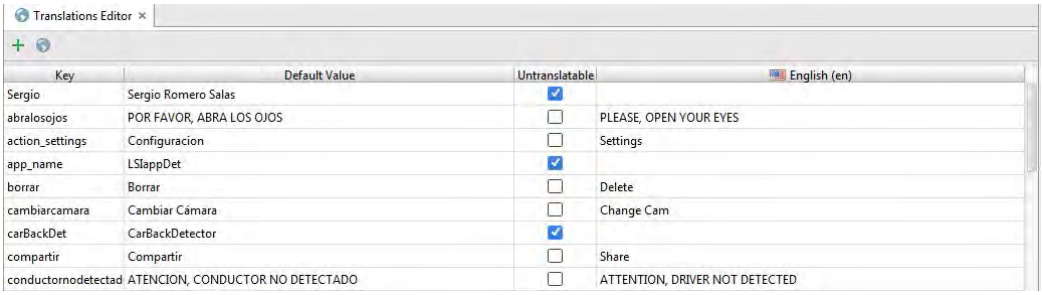

Figura 2.2: Ventana de la herramienta Translations Editor de Android Studio.

- Para finalizar, en la carpeta jniLibs podemos encontrar las bibliotecas nativas de la aplicación, en archivos preconstruidos con extensión so organizados en subcarpetas.

También podemos observar que paralelamente a la carpeta de la aplicación (app) está la carpeta openCVLibrary2411, en la cual se almacenan todas las funciones que nos ofrece la biblioteca de visión por computador OpenCV que analizaremos posteriormente. Además, en la carpeta Gradle Scripts se encuentra toda la información necesaria para la compilación de la aplicación.

2.1.3. Entorno de programación y estructura de ficheros

Una vez vista la forma en la que se estructuran los diferentes archivos de la aplicación y los más útiles para el desarrollo, pasamos a analizar los diferentes entornos de programación que nos ofrece Android Studio para cada tipo de archivo.

Los archivos Java son los más importantes ya que son los archivos en los que se encuentran los algoritmos programados de la aplicación. Para este tipo de archivos, Android Studio nos ofrece un entorno de programación típico de un editor de código, ayudándonos con aspectos como la tabulación automática, los colores para distintos tipos de variables o las sugerencias durante la escritura (Fig. 2.3).

La estructura típica de un programa en Java es la siguiente:

- En la parte superior nos encontramos con la definición del paquete (carpeta) donde se encuentra y los archivos importados de las librerías. Estos archivos se irán actualizando automáticamente según vayamos añadiendo elementos de esa librería a nuestro programa.
- Posteriormente podemos observar la declaración de la clase con el nombre de la actividad que queremos crear.

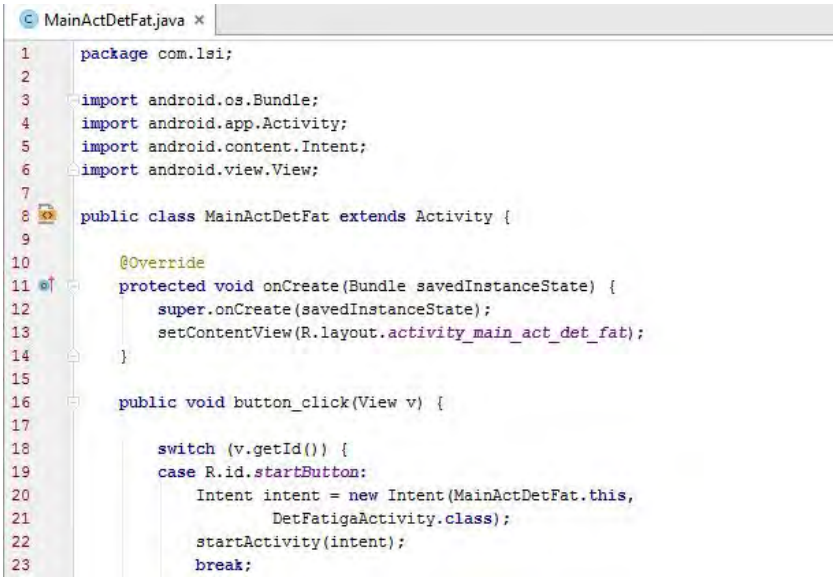

Figura 2.3: Editor de archivos Java en Android Studio.

- Dentro del cuerpo de la clase debemos definir las funciones que forman parte de nuestro programa. Para llevar un cierto orden, es recomendable declarar las variables externas a las funciones justo debajo de la declaración de la clase y, posteriormente, las funciones. Hay funciones que siempre deben ser implementadas para el correcto funcionamiento de la aplicación, como la función OnCreate que se ejecuta al iniciar la actividad y en la cual se debe llamar al xml correspondiente para ser mostrado. Además, dependiendo del tipo de actividad que se esté implementando, habrá otras funciones que deberán ser desarrolladas, como por ejemplo la función onCameraFrame cuando trabajemos con la cámara.

Para los archivos xml la disposición del entorno de programación de Android Studio es distinta que para los archivos Java. Al abrir un xml nos encontramos con una vista de programación junto a una vista previa del elemento (en los que se pueda mostrar) (Fig 2.4).

Además, en los xml que forman un layout, Android Studio nos ofrece, aparte de la vista comentada anteriormente, una interfaz para el diseño de los mismos que resulta muy útil e intuitiva (Fig. 2.5).

En esta interfaz podemos observar diferentes partes:

- En la parte central podemos observar una vista previa de la pantalla, pudiendo seleccionar el dispositivo en el que deseamos previsualizarla.

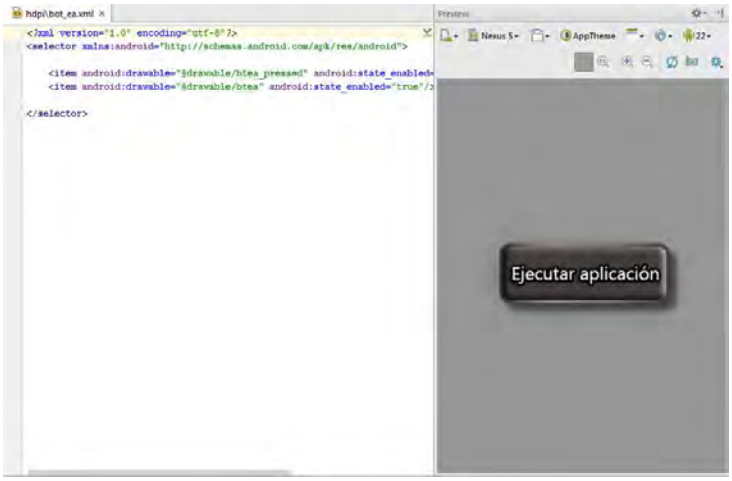

Figura 2.4: Edición de xml en Android Studio.

Además, en esta vista previa también podemos arrastrar elementos y colocar los elementos del layout por toda la pantalla.

- En la parte izquierda podemos observar los diferentes elementos que podemos agregar a nuestro layout. Entre ellos podemos encontrar botones, texto, contenedores de texto, barras deslizantes, imágenes, otros layout que se colocan dentro del principal para organizar los elementos, etc.
- Por último, en la parte derecha nos encontramos con un administrador de los elementos que hemos incorporado y más abajo con una ventana en la cual podemos modificar las propiedades de cada elemento. Estas propiedades dependen del tipo de componente y se puede tratar del texto que aparezca en un botón (los textos se obtienen a partir del archivo strings.xml), del estilo, color o fondo, del id del elemento por el cual en el archivo Java se le identificará para acceder a sus parámetros, de la función que se ejecutará al pulsarlo, etc.

Además, es interesante destacar que Android Studio nos ofrece una herramienta para dejar tareas pendientes durante la programación de la aplicación. Si durante nuestro desarrollo queremos dejar marcado una zona en la que nos falta código por implementar, con un comentario incluyendo la palabra clave "TODO"queda resaltado y, posteriormente en la barra inferior en la pestaña TODO podemos observar un resumen de dichas tareas.

2.1.4. Ejecución y depuración

Android Studio nos ofrece dos métodos mediante los cuales ejecutar nuestra aplicación para poder probarla mientras que la depuramos en busca de

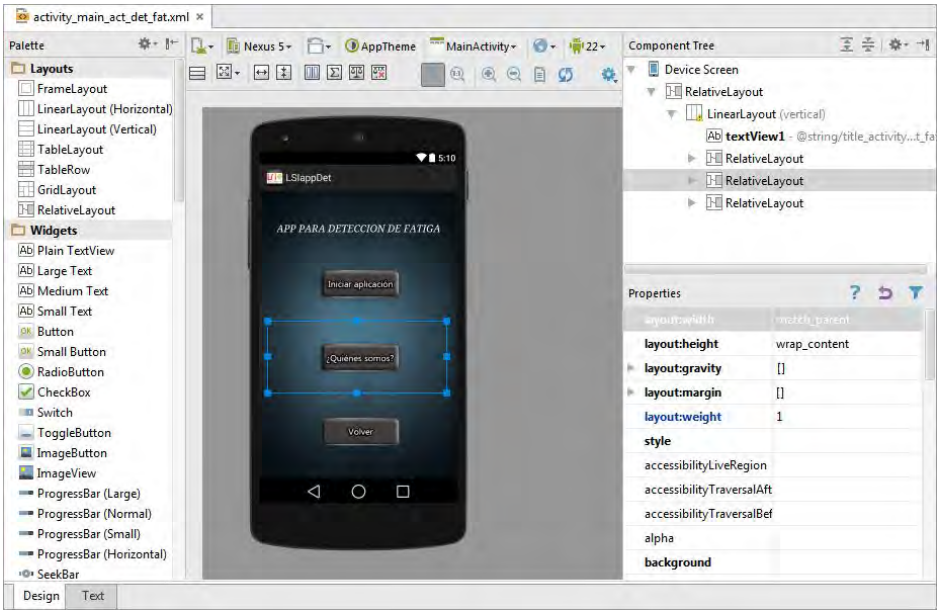

Figura 2.5: Interfaz de diseño de layout en Android Studio.

posibles fallos. (Fig. 2.6)

El primer método por el que se puede ejecutar nuestra aplicación es el emulador. Este método tiene la ventaja de no necestar un dispositivo móvil Android para ejecutarlo, pero la desventaja de una mayor complicación.

Para ello debemos crear un emulador con las características que queramos, como puede ser la versión de Android, el modelo del dispositivo, la versión de API (interfaz de programación de aplicaciones), etc. Una vez creado, al compilar y ejecutar la aplicación, seleccionamos el emulador y se abrirá en una ventana (Fig. 2.6). Debido a que en nuestra aplicación queremos usar la cámara del dispositivo, debemos que configurar el emulador para que use la webcam del ordenador que estemos usando.

Para evitar todos estos ajustes y simplificar la simulación, se ha escogido la opción de ejecutar la aplicación en un dispositivo real, en concreto el Smartphone Xperia M C1905, cuyas características se explicarán posteriormente.

Para ejecutar la aplicación en un dispositivo real, debemos seleccionar la primera opción en la pantalla anteriormente mencionada y seleccionar el dispositivo en el que la queremos simular, previamente habiendo instalado su driver en el ordenador.

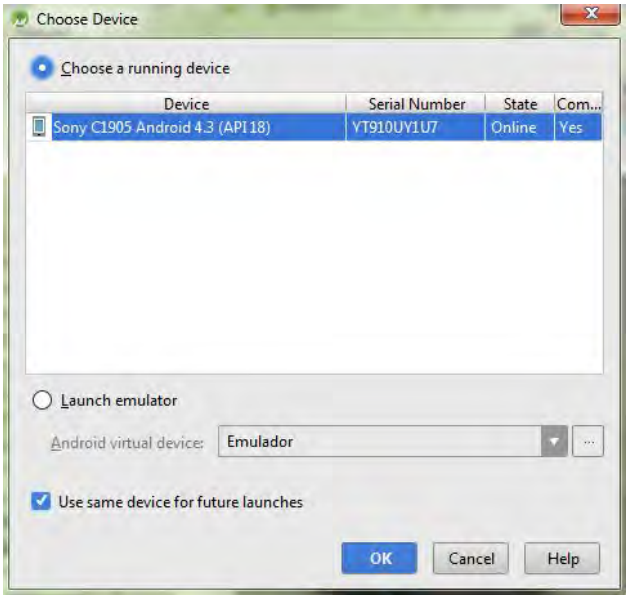

Figura 2.6: Modos de ejecución de una aplicación en Android Studio.

En cuanto a la depuración del programa, durante todo el proceso de compilación de la aplicación en la consola que aparece en la parte inferior de la derecha nos informará de qué tarea se está llevando a cabo y, al finalizar, los resultados con los errores y dónde se encuentran.

Una vez arrancada la aplicación podremos observar en la pestaña Logcat en la ventana debug toda la actividad de la aplicación (Fig. 2.7). Además, es posible programar avisos de eventos de la aplicación que queremos controlar para que aparezcan en esta ventana, así como ejecutar la aplicación paso por paso o introducir puntos de ruptura (BreakPoints).

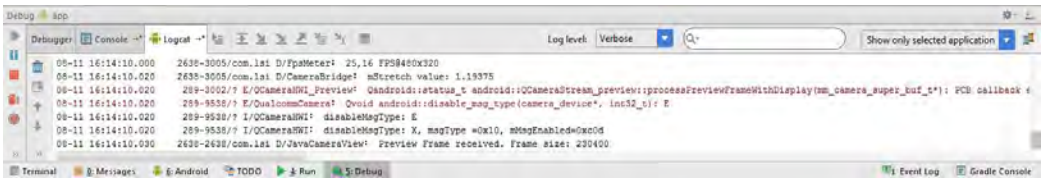

Figura 2.7: Pestaña Logcat para depuración de aplicaciones.

Con todas estas herramientas Android Studio nos facilita la labor de programar, ejecutar y depurar nuestra aplicación.

2.2. Librería OpenCV

Para el desarrollo de la parte de visión por computador presente en el proyecto se utilizará la librería de funciones abierta y gratuita OpenCV, en su versión de desarrollo para Android. Esta librería, originalmente desarrollada por Intel, se utiliza para el tratamiento de imágenes con el fin de analizarlas y obtener información a partir de ellas. Se trata de una librería multiplataforma presente para el desarrollo en Android, IOS, Windows, etc. En nuestro caso, la utilizaremos para el reconocimiento facial del conductor con el fin de poder detectar la fatiga en el mismo, trabajando con la versión 2.4.11 para Android.

La librería OpenCV nos ofrece multitud de recursos, entre los cuales se destacan las siguientes finalidades[26].

- Trabajo con imágenes procedentes de diferentes fuentes.
- Funciones para procesamiento y transformación de imágenes como ecualización, filtros, convoluciones, etc.
- Creación de histogramas en imágenes a color y en blanco y negro. Cambio de tipo de imagen (RGB, escala de grises, etc.).
- Funciones orientadas a la detección de bordes o contornos.
- Búsqueda de objetos y formas en la imagen.
- Control del movimiento de elementos de la imagen.

2.3. Lenguaje Java

En cuanto a lenguaje de programación, el lenguaje que se utiliza en este sistema operativo y por lo tanto en el trabajo es el lenguaje Java. Este lenguaje es un lenguaje concurrente orientado a objetos y diseñado para poder ser ejecutado en cualquier dispositivo (WORA (Write Once, Run Anywhere)). El código, una vez implementado, únicamente debe ser recompilado para poder ser ejecutado en otro dispositivo.

En general, la principal ventaja de Java como lenguaje de programación es su presencia en multitud de dispositivos como teléfonos móviles, procesadores remotos, microcontroladores, sensores, productos de consumo, módulos inalámbricos, etc. Además, Java también nos ofrece la posibilidad de ser ejecutado en un navegador web o desarrollar aplicaciones para foros online, encuestas, tiendas, etc[10].

Java deriva en gran parte de lenguajes de programación orientados a objetos como C o C++. A diferencia de estos lenguajes, Java está completamente orientado a objetos, estando todos sus elementos contenidos en una clase. Estas clases se encuentran en archivos .java con el mismo nombre que la clase.

Otros aspectos típicos de un programa como son los tipos de datos (int, double, string, boolean, etc.), las estructuras de control (if, for, while, etc.), los operadores (+, -, =, etc.), la declaración de variables y modificadores o la inclusión de archivos son heredados casi en su totalidad del lenguaje C++.

2.4. Dispositivo Móvil Sony Xperia M C1905

El dispositivo en el cual se realizarán las pruebas de la aplicación será un dispositivo móvil de gama media, en concreto el Smartphone Sony Xperia M C1905 (Fig. 2.8).

Figura 2.8: Smartphone Sony Xperia M C1905 [22].

Las características técnicas más importantes de este dispositivo son las siguientes[22]:

- Pantalla multitáctil capacitiva de 4 pulgadas.
- 115 gramos de peso.
- Procesador Qualcomm de doble núcleo (1 GHz).
- Versión de Android: Google Android 4.3 Jelly Bean.

- Cámara digital de 5 megapíxeles con zoom digital de 4 aumentos, flash LED y enfoque automático.
- Cámara frontal VGA.
- 1 GB de memoria RAM y 4 GB de memoria flash.

Una vez desarrollada la aplicación, debemos instalar los drivers del dispositivo para que sea reconocido por el ordenador, bien descargándolos desde la página oficial del fabricante o dejando a Windows que lo instale automáticamente.

Cuando los drivers estén instalados, debemos configurar el Smartphone para que permita instalar aplicaciones ajenas y activar el modo depuración por USB de aplicaciones. Así, Android Studio reconocerá nuestro dispositivo a la hora de ejecutar la aplicación, y la instalará para que se pueda ejecutar en el mismo.

Un detalle importante a tener en cuenta es que todo dispositivo aquel en el que queramos ejecutar una aplicación que use la librería OpenCV, como es nuestro caso, debe instalar previamente la aplicación OpenCV Manager disponible en Google Play (Fig. 2.9). Además, la versión de OpenCV que esté instalada en el dispositivo debe ser mayor o igual que la versión de la librería que hemos utilizado en la aplicación, por lo cual es importante comprobar si hay actualizaciones disponibles de la aplicación de OpenCV frecuentemente.

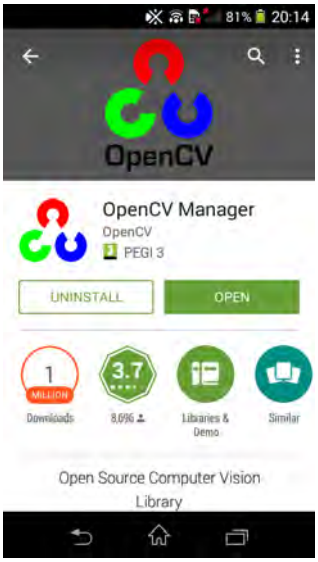

Figura 2.9: Aplicación OpenCV en Google Play.

Capítulo 3

Desarrollo de la aplicación

En este capítulo en primer lugar llevaremos a cabo una explicación de las fases y procedimientos más importantes llevados a cabo para la consecución de unos hitos hasta llegar al desarrollo de la aplicación de detección de fatiga que cumplimentara los requisitos más básicos.

Posteriormente pasaremos a analizar las mejoras implementadas sobre esta aplicación básica y, por último, expondremos los resultados finales.

3.1. Desarrollo de la aplicación básica

3.1.1. Estructuración básica de la aplicación en layouts y actividades

Para comenzar con la implementación de la aplicación, el primer paso fue analizar la estructura de la aplicación ya desarrollada para el LSI de la UC3M para poder añadir nuestra aportación a la misma . Esta aplicación, con el nombre de LSIappDet se desarrolló con el objetivo de servir como base para incluir futuras funcionalidades.

La aplicación tiene una pantalla (layout) principal que se abre al iniciar la misma (activity_main.xml, Fig. 3.1), en la cual podemos elegir entre ejecutar una aplicación del LSI, acceder a los ejemplos que vienen con la librería OpenCV o bien mostrar la información de desarrolladores (¿Quiénes Somos?). Además, si pinchamos sobre los iconos del LSI o de la UC3M nos redirigirá a sus páginas webs respectivamente.

En este layout no debemos modificar nada para implementar nuestra nueva funcionalidad, luego pasamos al layout que obtendremos si pinchamos sobre el botón “Aplicaciones LSI”.

Figura 3.1: Pantalla principal de la aplicación LSIappDet (activity_main.xml).

Este nuevo layout (activity_apps_lsi.xml, Fig. 3.2) está provisto de 2 botones, "Ejecutar aplicación" y "Volver". Mediante el primer botón saltará una ventana emergente con las diferentes aplicaciones o funcionalidades disponibles, mientras que al pulsar sobre el segundo botón volveremos a la pantalla inicial.

Esta pantalla sí debe ser modificada, debido a que debemos añadir nuestra aplicación a la lista de funcionalidades existentes. Debemos por lo tanto modificar el archivo .java correspondiente a este layout (AppsLSI.java) para que nuestra funcionalidad aparezca en esta lista y, además, para que al pulsar sobre ella accedamos a la actividad correspondiente.

Debemos crear un nuevo archivo Java para el menú principal de nuestra funcionalidad (MainActDetFat) y debe ser declarado en el AndroidManifest.xml. En esta declaración debemos incluir el nombre de la etiqueta, que será el nombre que encontremos en la ventana mientras esta actividad se esté ejecutando[1], en nuestro caso hemos elegido "Detección De Fatiga", valor

[1]En las capturas presentes en este documento puede que no aparezcan estos nombres ya algunas son

Figura 3.2: Pantalla para elegir aplicación (activity_apps_lsi.xml).

que hemos guardado en el string con la referencia inicioDetFat en el archivo strings.xml. Este procedimiento se deberá llevar a cabo con cualquier nuevo archivo java con su respectiva clase que se añada al proyecto.

Una vez creado el algoritmo para proceder a la funcionalidad de detección de la fatiga, procedemos a implementarla. Hemos optado, siguiendo la línea general del resto de la aplicación, por una ventana principal (activity_main_act_det_fat.xml, Fig 3.3) desde la cual se puede acceder a la actividad de la detección de la fatiga, a la información del desarrollador o volver a la ventana anterior.

Observando el layout de esta pantalla podemos ver que está compuesto por 3 botones y un textView, en el cual se encuentra la referencia al string que contiene el texto del título de la funcionalidad. Esta pantalla nos puede dirigir a 3 distintas actividades, a la pantalla de elección de aplicación si pulsamos el botón volver, la cual ya hemos explicado, a la ventana de la aplicación en sí, la cual explicaremos en el siguiente apartado, o la ventana

imágenes obtenidas desde la previsualización de Android Studio. Al ejecutar la aplicación en el dispositivo real sí aparecerían.

Figura 3.3: Layout de la pantalla principal de la funcionalidad de detección de la fatiga (activity_main_act_det_fat.xml).

de información del desarrollador (activity_qs_det_fat.xml, Fig. 3.4), la cual pasamos a explicar.

El layout de esta actividad(QSDetFat) es muy simple, ya que únicamente tenemos texto y no debemos implementar ningún algoritmo para ningún botón. El archivo xml consta de 7 textView cuyo contenido son strings previamente definidos en el archivos strings.xml. Estos textView están contenidos en layouts con el objetivo de tener una organización. No profundizaremos mucho en esta actividad, para mayor información observar el código completo en el anexo.

En los próximos capítulos nos centraremos en todo el algoritmo de la clase DetFatActivity, en el cual se implementa el código propio de la funcionalidad de la detección de fatiga.

3.1.2. Visión por computador y representación por pantalla

La primera meta propuesta fue la de representar una imagen obtenida de la cámara por pantalla, para posteriormente analizarla y modificarla mostrando una alarma cuando se vieran síntomas de fatiga. En este proyecto se

Figura 3.4: Layout de la pantalla de información de desarrollador (activity_qs_det_fat.xml).

tomarán como síntomas de fatiga los ojos cerrados.

Para comenzar hemos creado el layout correspondiente a esta actividad (activity_fat), el cual se compone únicamente de un componente del tipo JavaCameraView, definido en la biblioteca OpenCV, que será donde mostraremos la cámara del móvil. En su definición en el archivo activity_fat.xml debemos definir, entre otras propiedades, la cámara frontal o trasera o si queremos que se muestren los fotogramas por segundo (fps), que en nuestro caso utilizaríamos la cámara frontal y queremos mostrar los fps para poder controlar la velocidad de nuestra aplicación.

Cabe resaltar que esta actividad ha sido diseñada para trabajar en orientación horizontal (landscape), declarado así en el archivo AndroidManifest.xml, concretamente en la declaración del archivo java de la actividad.

Una vez explicado el.xml de la actividad, pasamos a la clase java de la misma ("DetFatigaActivity"). Después de la declaración, debemos definir las variables globales que vayamos a usar en la actividad, para llevar una buena estructuración. Trabajaremos sobre todo con la variable tipo Mat, que es la

que nos ofrece la librería OpenCV para guardar en ella la imagen y tratarla para obtener los datos de la misma.

Al tener la librería OpenCV instalada en otra aplicación y no en la que nosotros desarrollamos, tenemos la ventaja de reducir el peso de nuestra aplicación, pero también la desventaja de tener que inicializarla. OpenCV nos da las pautas para ello, declarando un objeto de la clase BaseLoaderCallback[16] que cargará la librería y habilitará la vista de la pantalla al finalizar esta carga.

La lógica que sigue esta inicialización (Fig. 3.5) es en primer lugar buscar la aplicación OpenCV Manager y si no se encuentra mandar un aviso para que sea instalada. Posteriormente, buscará si la librería no está instalada, en cuyo caso la instalará, inicializará y arrancará la actividad en la que se encuentre, habilitando la vista de la pantalla.

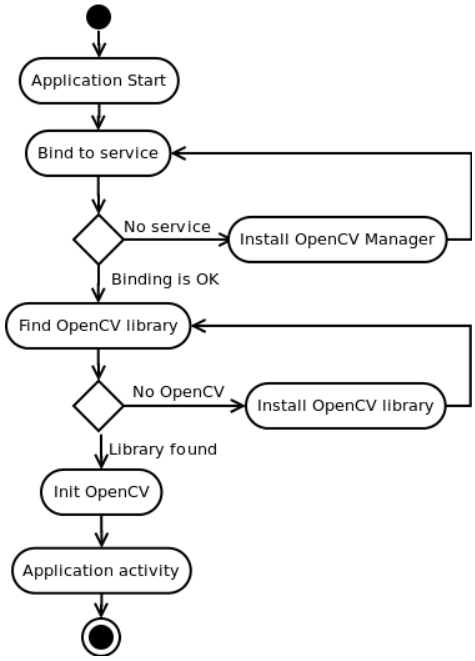

Figura 3.5: Lógica de la inicialización de la librería OpenCV con BaseLoaderCallback[16].

Para que la librería también se cargue cuando resumamos la aplicación, no solo cuando la creemos, debemos sobrecargar la función onResume para dicho fin. Además, también debemos sobrecargar las funciones onPause y onDestroy para que se deshabilite la vista cuando la aplicación se pare, se apague la pantalla, etc [2].

[2]Para mayor información consultar el código completo en el anexo

Continuando con el desarrollo de la aplicación, también vamos a sobrecargar las funciones onCameraViewStarted y onCameraViewStopped, con el objetivo de que cuando la cámara arranque inicialicemos la variable declarada anteriormente para guardar la imagen del frame y al parar la cámara, esta variable se libere. Con esto aumentamos la eficiencia de la aplicación, disminuyendo el uso de la pila (heap) de la misma.

Por último, antes de ponernos con la sobrecarga de la función onCameraFrame en la cual se vuelca el grueso del código, también debemos sobrescribir la función onCreate, función que es ejecutada cada vez que arranca la actividad. En dicha función debemos añadir al código funcional original de la función un flag para que la ventana se mantenga encendida mientras se esté ejecutando nuestra actividad, además de obtener los datos del elemento del tipo JavaCameraView creado en el layout de la actividad y de mostrar por pantalla el contenido del layout.

La función en la que se implementará la mayor parte del código de nuestra aplicación sera la función onCameraFrame, por lo tanto esta función también debe ser sobrecargada. Para el primer ejemplo de prueba en el que solo queremos mostrar por pantalla la imagen de la cámara frontal, el código es bastante sencillo. Esta función se ejecuta cada vez que llega un frame a nuestra cámara, y en ella debemos guardar este frame en escala de grises en nuestra variable interna de tipo Mat definida con anterioridad. El retorno de esta función será lo que se muestre por pantalla y debe de ser tipo Mat. En nuestro primer ejemplo únicamente debemos devolver la variable que hemos creado que contiene la imagen del frame para que sea mostrada por pantalla. Esta variable en este ejemplo se podría suprimir, pero es interesante tenerla para futuros desarrollos, ya que será la variable que modificaremos y mostraremos por pantalla.

Los resultados de este algoritmo inicial los podemos observar en la Fig. 3.6 [3].

Continuando con nuestro desarrollo de la funcionalidad, el siguiente paso sería intentar encontrar en la imagen los ojos del conductor, usando para ello las herramientas que nos provee OpenCV. Entre los métodos de reconocimiento de imágenes que nos provee OpenCV, utilizaremos el clasificador CascadeClassifier[18]. El funcionamiento de este clasificador en cascada se ba-

[3]Podemos observar que en este caso, al ser una captura de pantalla de la aplicación ejecutada en el dispositivo móvil Xperia M C1905, en el título de la aplicación nos encontramos con el que definimos en su declaración en el archivo AndroidManifest.xml.

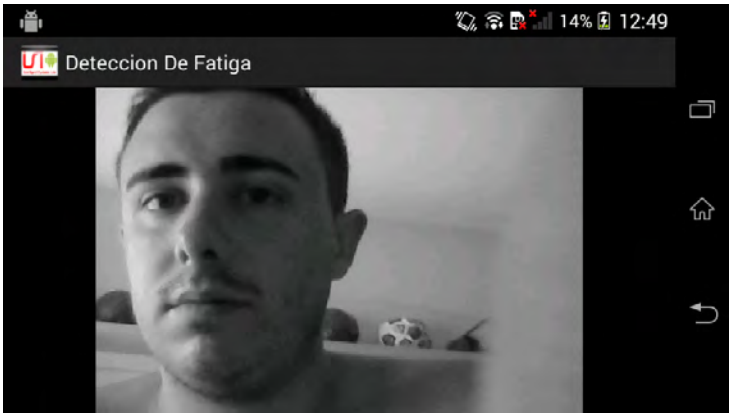

Figura 3.6: Resultado de la aplicación básica para mostrar por pantalla la imagen de la cámara frontal.

sa en el entrenamiento con imágenes positivas e imágenes negativas del objeto que queremos buscar en la imagen[20], extracción de las características más recurrentes de los mismos y su posterior comparación con la imagen y sus características, en forma de cascada, para dar mayor precisión y rendimiento (ejemplo del uso de clasificador en cascada para detección facial, Fig. 3.7).

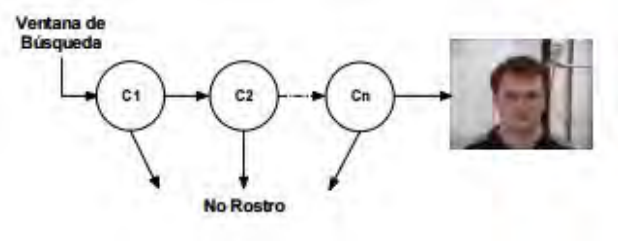

Figura 3.7: Clasificador en cascada para detección facial siendo C1, C2, ..., Cn cada una de las características[8].

Después del entrenamiento obtenemos un archivo .xml, el cual en nuestro caso nos lo ofrece OpenCV por defecto ("haarcascade_eye.xml")[2] entrenado con imágenes en escala de grises, sin necesidad de realizar el entrenamiento previo. Debemos copiar este archivo de la carpeta de la librería OpenCV a la carpeta de nuestra aplicación "raw".

Posteriormente, en la carga de la librería OpenCV, en el caso de que se haya cargado correctamente la librería, debemos incluir el código que nos permite cargar este archivo xml entrenado para la clasificación.

Una vez cargado el archivo previamente entrenado a nuestra actividad,

volvemos a pasar a la función onCameraFrame. Es recomendable para reducir el error del método de detección realizar una ecualización del histograma a la imagen en escala de grises previamente[25]. Esto consiste en tener una distribución más uniforme del histograma, es decir, tener una mayor equidad en la cantidad de píxeles con distintos niveles de gris (Fig. 3.8). Para ello usaremos la función que nos ofrece la librería OpenCV equalizeHist.

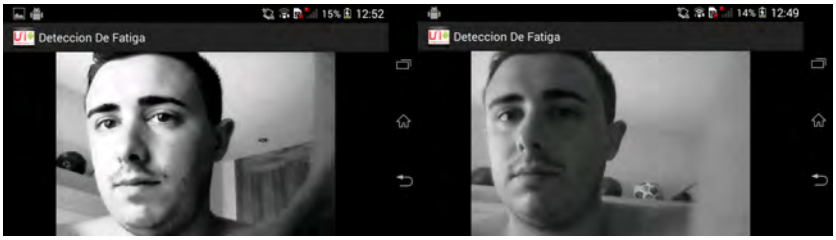

Figura 3.8: Diferencia entre imagen de salida ecualizada (izquierda) e imagen de salida original (derecha).

Para implementar la búsqueda de los ojos en la imagen debemos tener en cuenta varios aspectos.

- En primer lugar resaltar que, aunque realicemos la búsqueda de la posición de los ojos en la imagen en escala de grises, la imagen que modificaremos y representaremos será en color. Ambas imágenes tienen las mismas dimensiones, lo cual nos hace más sencillo trabajar con las dos a la vez.
- La función detectMultiScale de la clase CascadeClassifier[18] sirve para detectar múltiples objetos en la imagen y es la que utilizaremos para nuestra búsqueda. Los atributos son los siguientes:
  - Como primer parámetro, le pasaremos la variable tipo Mat que representa la imagen en la que queremos buscar el objeto.
  - El segundo parámetro es donde se guardarán los resultados, el cual es de tipo MatOfRect, ya que guardará los resultados como la posición de un rectángulo en la imagen que contenga al objeto en sí.
  - El siguiente es el valor del tamaño de la escala, que define cuánto se va a reducir la imagen, por el cual tomamos el valor por defecto 1.1.
  - El cuarto parámetro es un parámetros muy importante, es el mínimo número de vecinos próximos que debe encontrar el clasificador para confirmar ese objeto como válido. Al aumentar este número aumentamos la precisión y viceversa.

- Para continuar, el siguiente parámetro es un flag que no se utiliza para nuevos clasificadores, por lo tanto dejamos un 0.
- Por último, le pasaremos a la función dos parámetros que nos indican el mínimo y el máximo tamaño de objeto que hay que tener en cuenta. Fuera de estos límites, no se dará como válido, en este caso los dejaremos en blanco.

- Para cada uno de los resultados dibujaremos un círculo de color rojo alrededor del ojo. Cabe destacar que el color ya se define como un escalar de tres valores enteros, en el cual cada uno de ellos representa el color rojo, verde y azul, variando de 0 a 255. En el caso del color rojo es (255,0,0).

En cuanto al resultado de este método (Fig. 3.9) podemos observar que se dan lugar muchos falsos positivos, además de que la aplicación se ejecuta muy lentamente (alrededor de 1,2 fps). Una solución al primer problema sería aumentar el mínimo número de vecinos próximos para aumentar la precisión[11], pero esto no solucionaría el problema de la lentitud de la aplicación ya que seguiríamos buscando muchos objetos en una región muy amplia.

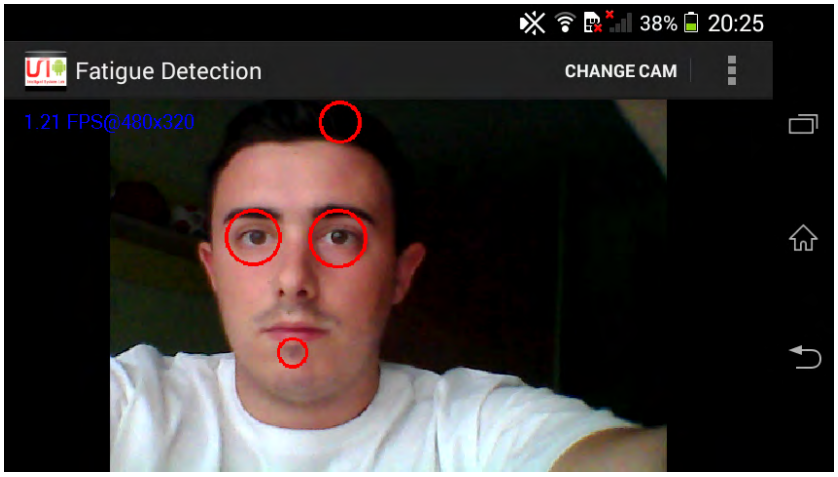

Figura 3.9: Resultado de la búsqueda por toda la imagen de ojos usando un clasificador en cascada.

Para solucionar este problema, la mejor solución es disminuir la región de interés donde buscar en vez de buscar los ojos por toda la imagen. Esta región de interés se creará buscando la cara del conductor y, posteriormente, reduciendo a la parte en la que se encuentran los ojos.

Para ello, debemos crear otro clasificador que busque en la imagen la cara del conductor. Afortunadamente, OpenCV también nos provee de xml pre-

viamente entrenado para la búsqueda de rostro mediante un clasificador en cascada. Por lo tanto debemos definir esta variable y cargar el archivo como hicimos con el necesario para detectar los ojos.

Además, en este caso vamos a definir un tamaño mínimo de la cara que queremos encontrar. Vamos a utilizar un factor relativo al tamaño máximo de la imagen de un 30 % de la misma[4], tamaño por debajo del cual no reconoceremos ninguna cara. Pasaremos de este valor relativo a un valor absoluto multiplicándolo por el número de filas de la imagen.

Se pintará un rectángulo en el rostro y otro en la región de interés (Fig. 3.10). Posteriormente se creará otra Mat que contenga la región de interés y en ella se buscarán los ojos del conductor. Hay que tener en cuenta que al pasar los puntos de los ojos a la imagen general para dibujar el círculo habrá que sumar en ambos ejes la posición del rectángulo de la cara respecto de la imagen total. En cada cara buscaremos los ojos en la región de interés y los pintaremos (suponemos más de una cara aunque a efectos prácticos solamente debería haber una). Para detectar correctamente cuándo el ojo está cerrado o abierto, aumentaremos la precisión del clasificador, aumentando el mínimo número de vecinos próximos. Además, añadiremos un texto en la parte inferior del rostro con la información sobre el número de ojos abiertos, obteniéndola a partir de la longitud del array de rectángulos de los ojos que nos devuelve la función detectMultiScale[18] como resultado de la búsqueda.

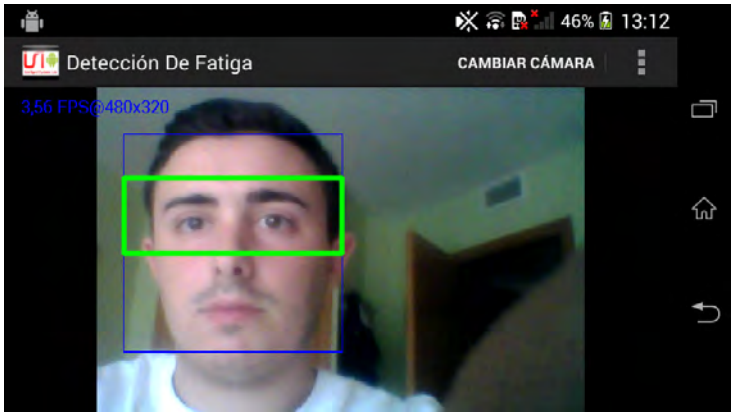

Figura 3.10: Representación de la región de interés (verde) en el rostro del conductor (azul).

Podemos observar que los resultados (Fig. 3.11) son mucho mejores que

[4]Este valor se podrá modificar (explicado en el apartado 3.2 Implementación de mejoras).

en el anterior caso, tanto en detección de ojos como en velocidad de la aplicación (en torno a 3,5 fps[5], alrededor del triple que en el caso anterior).

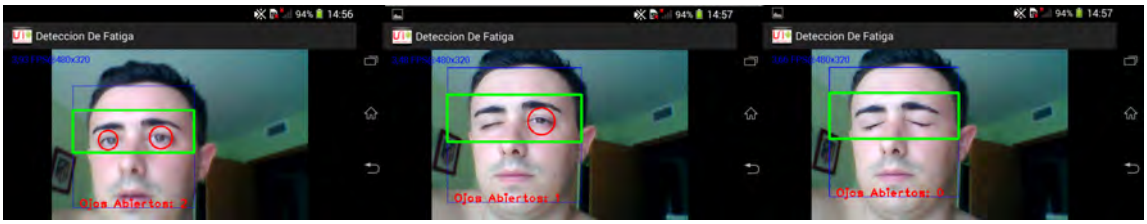

Figura 3.11: Resultado de la búsqueda de ojos reduciendo la región de interés

Con esta implementación podemos tener una base fiable para desarrollar una lógica para una alarma en el caso de que se detecte fatiga en el conductor.

3.1.3. Lógica y desarrollo de la alarma

La lógica para la activación de la alarma una vez obtenida la información de la imagen se puede ver explicada en el diagrama de flujo de la Fig. 3.12.

En este diagrama podemos observar que nos encontramos con dos estados. El estado 0 es el estado de reposo en el cual no hemos detectado ninguna amenaza, pero en el momento en el que se detectan menos de dos ojos abiertos, pasamos al estado 1. En este estado debemos comprobar si nuestra alerta es verdadera, ya que puede ser un simple parpadeo o un fallo en el reconocimiento de la imagen. Para concretar que es una falsa alarma y volver al estado 0 debemos detectar los 2 ojos abiertos en 3 frames consecutivos, así evitaremos errores en nuestra aplicación. Para poder detectar falsas alarmas damos un margen de tiempo de 3 segundos controlado por un temporizador iniciado al detectar el primer frame "defectuoso". Si, por lo contrario, seguimos sin detectar los dos ojos abiertos, se activará una alarma para la cual solo se puede salir con la detección de los 2 ojos abiertos.

En nuestra aplicación, este diagrama de flujo se controla mediante el código que se puede observar abajo definido en la función onCameraFrame.

[5]Estos fotogramas o frames por segundo dependen del dispositivo en el que se implemente y del procesador del mismo.

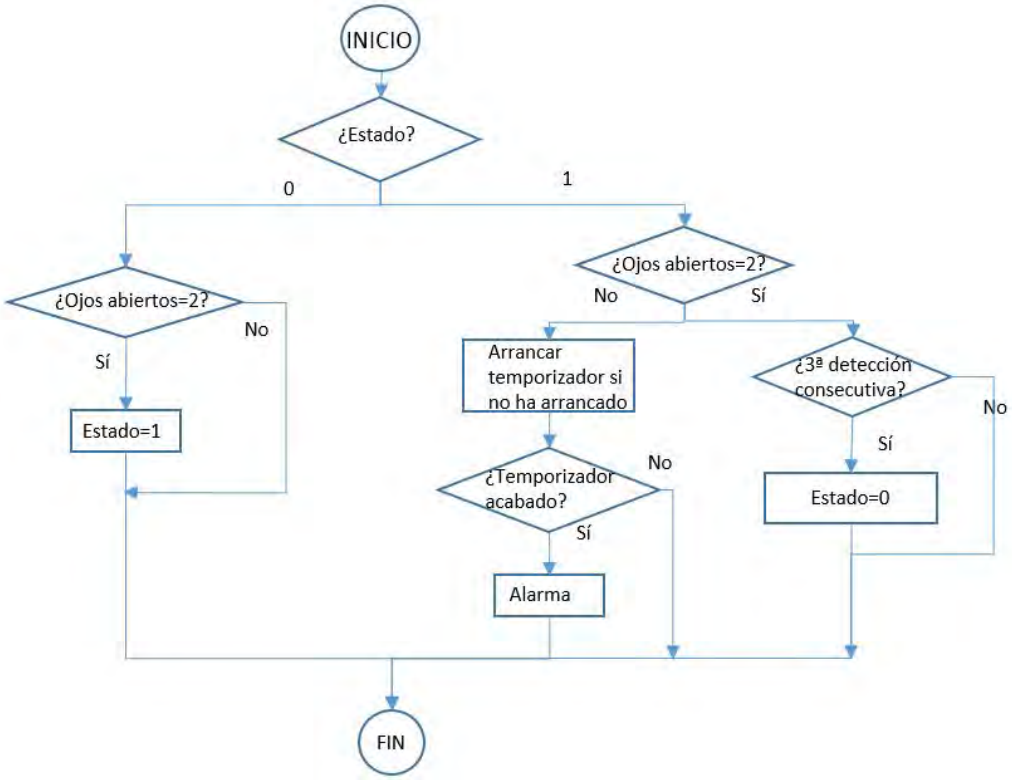

Figura 3.12: Diagrama de flujo para la activación de la alarma.

```
/**Código para la activación de la alarma implementado en la función
    onCameraFrame*/

        switch (estado){
           case 0:
              if(Ojosabiertos<2){
                 estado=1;
                 break;
              }
           case 1:
              if(Ojosabiertos==2){
                 cuentadetecciones++;
                 if(cuentadetecciones==3){
                    estado=0;
                    cuentadetecciones=0;
                    tempiniciado=0;
                    T.cancel();
                    cuentaacabada=0;
                 }
              }
              else{
                 if(tempiniciado==0){
```

```
22                      T.start();
23                      tempiniciado=1;
24                  }
25                  cuentadetecciones=0;
26                  if(cuentaacabada==1){
27                      Core.putText(rgb_img, "POR FAVOR, ABRA LOS OJOS
                            ", new Point
28                          (faceArray[i].tl().x,P2.y),1, 1.5,
                                EYE_CIRCLE_COLOR, 2);
29                      mire_carretera.start();
30                  }
31              }
32              break;
33          }
34          //Aviso de conductor no detectado
35              if(faceArray.length==0){      /**Variable a la que
                    pasamos los resultados de la detección de caras en
                     forma de array*/
36                  Core.putText(rgb_img, conductornodetectado,
37                        new Point(0,rgb_img.rows()/2), 1, 1.5,
                              EYE_CIRCLE_COLOR, 2);
38              }
```

Sobre este código hay varias cosas que resaltar.

- Las variables para la lógica de la alarma son todas tipo entero y estáticas con el objetivo de no perder su contenido cada vez que se ejecute la actividad, además de ser inicializadas a 0. La variable "estado" controla el estado en el que nos encontramos, la variable "cuentadetecciones" controla la cantidad de detecciones consecutivas de ambos ojos abiertos, la cual se reiniciará al encontrarnos con un frame que no cumpla esta condición, la variable "cuentaacabada"la usamos para controlar el fin del temporizador y, por último, la variable "tempiniciado"nos sirve para saber si el temporizador se ha iniciado y no volverlo a reiniciar en cada frame.
- En cuanto al temporizador, se trata de un temporizador que comienza en 3 segundos y va avanzando hasta llegar a 0. El primer parámetro que le pasamos es el tiempo con el que arrancará, y el segundo el intervalo en el que realizará la función onTick. Está diseñado para poder mostrar un mensaje o realizar alguna acción en esta función que se activa cada cierto tiempo, pero nosotros no usaremos esta funcionalidad, ya que solo nos interesa cuando llegue al final. Por lo tanto, sobrescribiremos la función onFinish para que active el flag cuentaacabada y podamos detectar cuando ha finalizado.
- La alarma consiste en un sonido emitido aconsejando al conductor un mayor atención a la carretera, mientras se muestra un mensaje por pantalla. Para el sonido debemos añadir la fuente en un formato legible

(en nuestro caso mire_a_la_carretera.mp3) en la carpeta raw de nuestra aplicación y, en la función onCreate, inicializar nuestra variable de tipo MediaPlayer con este archivo mp3.

- También hemos implementado un aviso por pantalla para el caso de que no se detecte el rostro del conductor, utilizando el parámetro "faceArray.length", que nos indica la longitud del array de las caras localizadas, es decir, el número de caras encontradas en la imagen.

Con este desarrollo y a falta de la implementación de mejoras para la aplicación, disponemos de una aplicación de funcionalidad básica para la detección de fatiga. Los resultados, como podemos observar en el siguiente vídeo, son bastante aceptables[6].

3.2. Implementación de mejoras

En este apartado explicaremos las mejoras implementadas sobre la aplicación de funcionalidad básica de detección de la fatiga.

[6]Podemos observar en los vídeos expuestos en este proyecto que los fps son menores debido a que, al iniciar en el dispositivo un proceso de grabado de la pantalla, consume recursos del procesador y disminuye los fotogramas procesados por segundo.

3.2.1. Alarma para síntomas de fatiga prolongados

Una mejora importante de la aplicación es la que corresponde con la implementación de una alarma para la detección de síntomas de fatiga prolongados. Esta alarma saltará cuando hayamos detectado en 3 ocasiones al menos síntomas de fatiga y nos recomendará que paremos el coche y realicemos un descanso de la conducción.

La variable que controla la lógica esta alarma es la variable "eventos", que define el número de ocasiones en el que se han detectado alarmas verdaderas, es decir, los ojos cerrados del conductor. Esta variable se debe definir como todas la demás que controlan la lógica de la primera alarma como variable global estática e inicializada con valor 0. Su incremento se producirá cada vez que pasemos del estado de alerta al estado de reposo abriendo los ojos. Por lo tanto debemos implementar el código que veremos abajo en el estado 1 y, dentro del mismo, con la condición de que los ojos estén abiertos. Además, para que no contabilice una falsa alarma, debemos poner como condición que el temporizador haya acabado y, para asegurarnos de que la alarma se ha superado, que el numero de detecciones es 3.

```
1 /**Extracto de la función onCameraFrame donde se implementa la
      lógica para la alarma a largo plazo*/
2
3    if(cuentaacabada==1 && cuentadetecciones==3){
4          eventos++;
5          if(eventos>2){
6             pare_descanse.start();
7          }
8    }
```

En cuanto a la alarma en sí, consiste como la anterior de un sonido y un mensaje por pantalla. Por lo tanto, debemos añadir otro archivo mp3 a nuestra carpeta raw con el mensaje que queremos escuchar y inicializar otra nueva variable (en este caso "pare_descanse") en la función onCreate exactamente igual que hicimos con anterioridad. Para que se muestre el mensaje por pantalla y se escuche el audio una vez el conductor haya abierto los ojos, es necesario iniciar el audio dentro del bucle del estado de la aplicación, pero, para que se muestre el texto, incluir el siguiente código fuera del bucle.

```
1 /**Extracto de la función onCameraFrame donde se implementa la
      lógica para la alarma a largo plazo*/
2
3    if(pare_descanse.isPlaying()){
4       Core.putText(rgb_img, "POR FAVOR, PARE Y DESCANSE", new Point(
            faceArray[i].tl().x,
5             P2.y), 1, 1.5, EYE_CIRCLE_COLOR, 2);
```

```
6     }
```

Así, mientras el audio se esté reproduciendo, se mostrará el mensaje por pantalla, aunque hayamos vuelto al estado de reposo.

3.2.2. Cambio del tamaño mínimo de rostro

La distancia a la que se encuentre el conductor del dispositivo en cuestión puede ser variable. En el caso de que este se encontrara demasiado lejos de la cámara, puede que su rostro no entre en el umbral detectable por la función detectMultiScale[18] del clasificador y no sea detectado.

Para solucionar este problema, se le ofrece al usuario una opción de configurar el tamaño mínimo de cara reconocible por la aplicación, disminuyendo este valor según se aleje de la lente de la cámara. Debemos desarrollar un menú en el cual demos varios valores prefijados (lo implementaremos para los valores de 20, 30 40 y 50 %) para que el usuario elija uno de ellos.

El primer paso es desarrollar el archivo xml del menú que vamos a desplegar en nuestra actividad. Para ello creamos un nuevo archivo en la carpeta menu denominado "det_fatiga.xml.[en] el cual declaramos 4 elementos ("items") que son las opciones que aparecen en el menú. Cada uno de los elementos del menú está definido por los siguientes 4 parámetros.

- El id del elemento, por el cual luego nos referiremos a él cuando queramos programar la acción ejecutada al pulsarlo.
- El segundo parámetro nos especifica la prioridad de aparición que tiene el elemento si se encontrara con otros del mismo tipo. Como en nuestro caso queremos que aparezcan según los hemos definido, lo dejamos todos en 0.
- El tercer parámetro nos define de qué manera aparecerá este elemento en la barra superior. En nuestro caso todos estarán definidos por el string "never", ya que los mostraremos en un menú desplegable.
- Por último, el título define el nombre que aparecerá en el botón.

Una vez definido el menú que vamos a desplegar, debemos implementar las acciones en la clase de nuestra actividad. Los aspectos a detallar de la creación e implementación del menú son los siguientes.

- La sobrescritura de la función onCreateOptionsMenu se realiza para mostrar por pantalla nuestro menú al arrancar la actividad con la función.

- Posteriormente, la programación de las acciones de cada botón se realiza en la función onOptionsItemSelected, llamando en cada caso a la función setMinFaceSize, creada totalmente por nosotros, pasando como parámetro el valor relativo en tanto por uno.
- Por último, esta función actualiza el valor de la variable global "mRelativeFaceSizez pone el valor absoluto a 0, para que, con el código implementado en la función onCameraFrame, se actualice este valor en el siguiente fotograma.

3.2.3. Cambio de cámara

Debido a que la cámara frontal normalmente suele disponer de menor calidad que la trasera o debido a motivos de colocación del dispositivo, una mejora interesante de la aplicación es permitir que el usuario de la misma realice la monitorización con la cámara que desee.

Para comenzar, debemos añadir otro elemento al menú creado con anterioridad para la selección del mínimo tamaño de cara. Este elemento, a diferencia de los anteriormente creados, en el parámetro "showAsAction"no está definido para ser mostrado en el menú desplegable ya que, por decisión de diseño, se ha decidido mostrar en la barra superior directamente.

Así, el menú de la actividad DetFatActivity queda definido en su totalidad (Fig. 3.13).

Figura 3.13: Menú para selección del tamaño de cara.

Continuamos programando el cambio de cámara. Cada vez que pulsemos este botón, la variable creada para controlar la cámara activa (mCameraId) cambiará de 1 a 0 y viceversa. Posteriormente, se deshabilitará la visión de

la cámara para cambiar el índice de la misma dependiendo de la variable mCameraId y se volverá a habilitar la vista, con la cámara ya cambiada.

3.2.4. Implementación para varios idiomas

La última mejora que se ha implementado en la aplicación ha sido el desarrollo de la misma para ser interpretada en varios idiomas, dependiendo del idioma en el que se encuentre el dispositivo. Como ya hemos comentado en la introducción, para la traducción se forman varios archivos de strings con un código de idioma para que el dispositivo acceda a ellos dependiendo del idioma presente.

Para facilitar el trabajo, Android Studio pone a nuestra disposición la herramienta "Translations Editor", con la cual podemos editar la traducción fácilmente sin tener que estar cambiando constantemente de archivo. Los strings como nombres o correos se pueden marcar como intraducibles, y aparecerán igual en cualquier idioma.

En concreto se ha realizado la traducción de la aplicación completa al inglés estadounidense, dejándose como opción por defecto el idioma español. Es importante y forma parte de las buenas prácticas de Android definir siempre el string en el archivo strings.xml y, posteriormente, referirse a él cuando sea necesario su uso. Para los textos que están creados como una referencia al xml del string, únicamente hay que traducirlos, mientras que para los textos que nos encontramos en las definiciones de las clases (como los textos que se presentan en la pantalla de los avisos de detección de fatiga) deben ser creados strings para ellos y inicializarlos en la clase.

Cabe destacar que los botones de la aplicación debido a que se trata de imágenes prediseñadas y los archivos de audio no se verán afectados a la traducción.

3.3. Resultado final

En el vídeo que se muestra abajo podemos observar los resultados de la aplicación tras la implementación de las mejoras explicadas con anterioridad. Podemos observar como el cambio de cámara se realiza correctamente tras pulsar el botón que encontramos en la barra superior. También podemos apreciar que se despliega el menú con las opciones para seleccionar el mínimo tamaño de rostro reconocido en la imagen. Por último, la alarma para los síntomas de fatiga prolongados funciona también correctamente, indicando al

conductor que pare y descanse tras mostrar, en 3 ocasiones o más, síntomas de fatiga.

Capítulo 4

Análisis de los resultados

En el siguiente apartado realizaremos un breve análisis de los resultados obtenidos tras la implementación de la aplicación. Para ello utilizaremos 4 secuencias de alrededor de 25 segundos cada una capturadas por el dispositivo móvil, con diferentes iluminaciones y utilizando una búsqueda por toda la pantalla o únicamente en las regiones de interés (ROI), con el fin de analizarlas y compararlas. Las secuencias se pueden observar en el siguiente vídeo.

Las secuencias utilizadas son las siguientes.

- La primera secuencia (arriba a la izquierda) se trata de una secuencia con una iluminación escasa y utilizando el método de búsqueda por toda la imagen de los ojos, sin tener en cuenta las regiones de interés.
- En cuanto a la segunda secuencia (arriba a la derecha) también se trata de una secuencia con baja iluminación, pero en este caso utilizamos el método de las regiones de interés.
- El tercer vídeo se trata de una secuencia con una mejor iluminación, pero sin utilizar el método de las regiones de interés.
- Para finalizar, abajo a la derecha en la cuarta secuencia, podemos observar una iluminación correcta y la utilización del método de las regiones de interés.

Además, es importante para este apartado explicar los conceptos de falso positivo y falso negativo. El primero ocurre cuando nuestro clasificador detecta un elemento, pero este no se encuentra en la imagen y, el segundo aparece cuando no lo detecta pero sí aparece en la imagen.

4.1. Presentación de resultados de las secuencias analizadas

Se ha realizado un análisis de todos los frames de cada una de las secuencias para detectar los falsos positivos y los falsos negativos que se encontraban en cada una, tanto de rostro como de ojos, cuyos resultados podemos observar en la siguiente tabla. En las secuencias donde no aplicamos el método de las ROI no corresponde analizar fallos en la detección del rostro.

			ROSTRO		OJOS	
	SECUENCIA	FRAMES	FALSO NEGATIVO	FALSO POSITIVO	FALSO NEGATIVO	FALSO POSITIVO
Mala iluminación sin ROI	1	7			0	2 (28,57 %)
Mala iluminación con ROI	2	69	25 (36,23 %)	0	2 (2,9 %)	0
Buena iluminación sin ROI	3	9			0	6 (66,67 %)
Buena iluminación con ROI	4	64	0	0	0	3 (4,69 %)

También se ha realizado un análisis de con qué frecuencia se han detectado los errores en cada una de las secuencias(Fig. 4.1).

Además, también se ha analizado el número de frames procesados en cada secuencia en los 25 segundos de duración (Fig. 4.2).

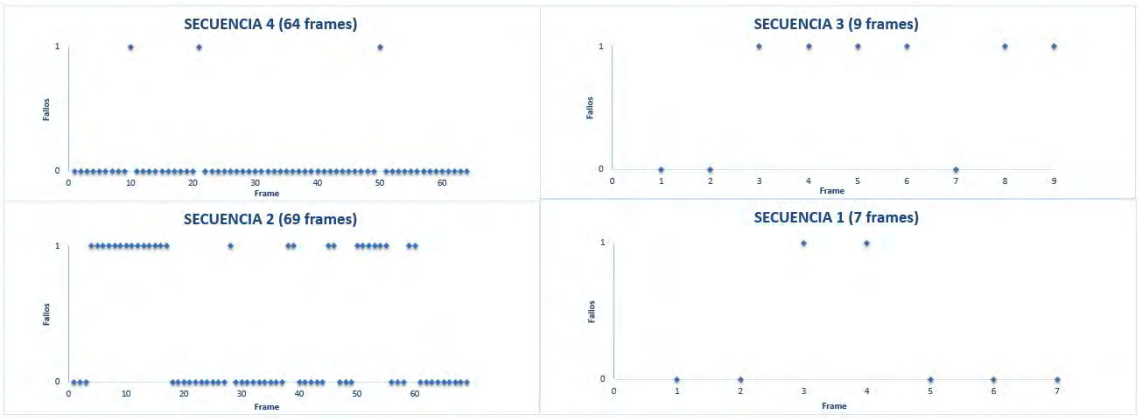

Figura 4.1: Aparición de los errores en las secuencias.

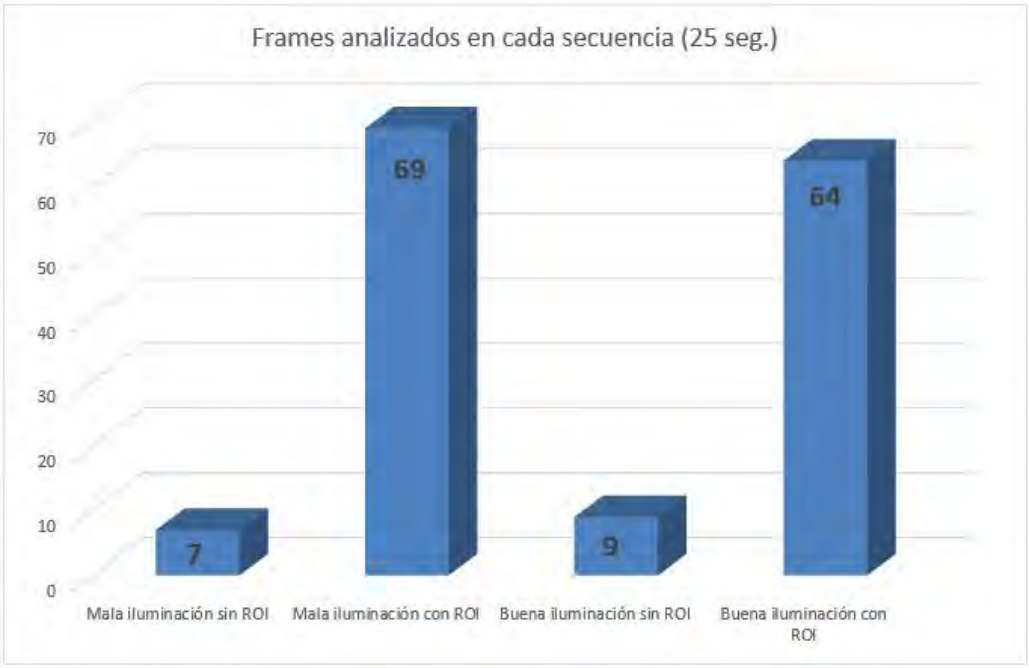

Figura 4.2: Frames procesados en cada secuencia.

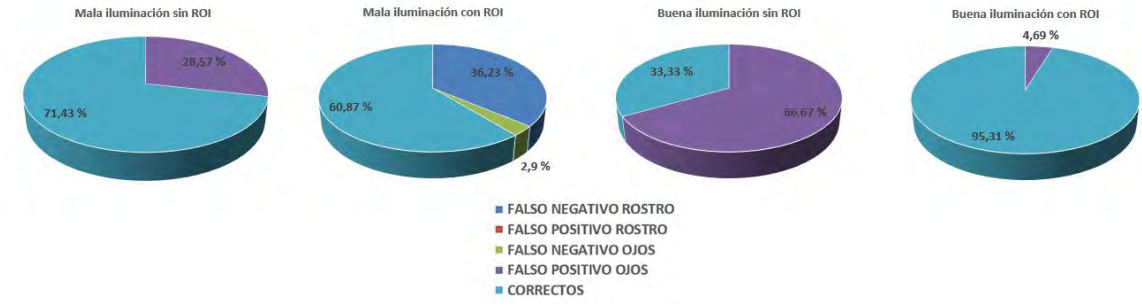

Figura 4.3: Detalle de los frames analizados de cada secuencia.

El último análisis que se ha realizado ha sido una comparación de, sobre cada secuencia, que tipo de fallo ha dado cada frame o si se ha realizado una correcta búsqueda sobre la imagen (Fig. 4.3).

4.2. Análisis de resultados

Realizaremos ahora un análisis de los resultados obtenidos y la extracción de conclusiones de los mismos.

- Para comenzar, como podemos observar en el gráfico de la imagen 4.2, las secuencias en las que se ha utilizado el método de las regiones de interés es capaz de analizar muchos más frames en el mismo tiempo, obteniendo así una funcionalidad más dinámica y eficiente. Esto es debido a que el clasificador en cascada de los ojos consume muchos recursos y, al reducir el área de trabajo únicamente a la región superior de la cara, reducimos los recursos empleados considerablemente para una mayor eficiencia.

- Continuamos analizando los fallos en el análisis de los fotogramas. Es destacable como, al aplicar las regiones de interés al análisis en imágenes con buena iluminación se reduce el número de fallos en las imágenes procesadas, sobre todo de falsos positivos de ojos, ya que, al buscar en toda la imagen, tenemos mayor probabilidad de encontrar falsos positivos de los mismos.

- En el caso de aplicar este método cuando la imagen tiene mala iluminación, no aumentan el número de frames correctamente analizados debido al aumento de los falsos negativos de rostros. Esto se debe a que el clasificador de las caras es más sensible a la mala iluminación como puede ser el de los ojos.

- También es interesante analizar el caso de la iluminación incorrecta de la imagen procesada por el algoritmo. Podemos observar que el ratio de frames procesados correctamente disminuye, aplicando el método ROI. Aunque esta disminución de la calidad sea significativa, es un factor menos determinante que la aplicación del método ROI.
- Para finalizar, es interesante observar que en el caso de la aplicación del método ROI con una iluminación correcta, encontramos pocos errores y con una separación en el tiempo importante, lo cual podemos despreciar al tener un filtro en el algoritmo que exige una detección de al menos 3 frames consecutivos para asignarla como correcta.

Capítulo 5

Costes del proyecto

En este capítulo analizaremos los costes del proyecto, exponiendo para comenzar las horas utilizadas en la realización del mismo y, posteriormente, desglosaremos todos los costes hasta llegar al presupuesto total del proyecto.

El tiempo utilizado por el ingeniero para el desarrollo del proyecto se explica en la tabla siguiente.

ETAPA	TIEMPO
Estudio teórico de la visión por computador	40 horas
Comprensión del trabajo a realizar y estudio de la aplicación del LSI	60 horas
Implementación del código de la aplicación	150 horas
Realización de pruebas del algoritmo	10 horas
Redacción de la memoria	110 horas
Total	**370 horas**

Así, con un total de 370 horas, pasamos a calcular el precio de la mano de obra del ingeniero suponiendo un salario del mismo de 20 € a la hora.

$$370\ horas\ x\ 20\ €/hora = 7.400€ \tag{5.1}$$

Continuamos con el desglose de los materiales empleados para la realización del proyecto. Entre ellos se encuentra el ordenador portátil Hp Pavilion Dv6 en el cual se ha implementado toda la aplicación, así como la redacción de la memoria, sobre el cual calcularemos el precio del periodo correspondiente al tiempo de uso del mismo. Supondremos una vida media de ordenador portátil de 5 años, realizando un uso del mismo de 4 meses durante el cual se ha implementado el proyecto, con un precio de 899 €.

$$4\ meses\ x\ 899\ €/(5\ años\ x\ 12\ meses/año) = 59,93\ € \tag{5.2}$$

Además, también debemos incluir el uso del dispositivo móvil Sony Xperia M C1905 con un precio de 119 €, una vida media de 15 meses[6] y un uso del mismo de 2 meses durante el tiempo en el que se realizaron las pruebas del algoritmo.

$$2\ meses\ x\ 119\ €/15\ meses = 15,87\ € \tag{5.3}$$

En la tabla siguiente podemos encontrar el cómputo global de todos los gastos del proyecto. Para concluir y como podemos observar en la tabla, el presupuesto total para el desarrollo del proyecto asciende a la cantidad de 7.475,8 €.

CONCEPTO	COSTE
Mano de obra del ingeniero	7.400 €
Ordenador portátil Hp Pavilion Dv6	59,93 €
Smartphone Sony Xperia M C1905	15,87€
Total	**7.475,8 €**

Capítulo 6

Conclusiones

Se presentan a continuación la discusión final del proyecto junto con un apartado para posibles desarrollos futuros.

6.1. Discusión final

Se han expuesto en este documento todos los procedimientos llevados a cabo para la creación de la funcionalidad de detección de fatiga para una aplicación en plataforma Android, tanto la programación en Android Studio como la parte de visión por computador usando las librerías OpenCV.

Para comenzar han sido descritas las herramientas utilizadas, así como se ha realizado una pequeña introducción a las herramientas que nos ofrece OpenCV para el tratamiento y análisis de imágenes.

Además, también se han analizado las posibles soluciones alternativas que se podrían implementar para la obtención de datos de imágenes y se ha realizado un análisis de los costes del proyecto.

Una vez finalizado el proyecto y tras realizar un breve análisis de los resultados, podemos concluir con que se ha implementado una aplicación robusta para la detección de la fatiga en un conductor tras una elección correcta de los recursos disponibles. Se contribuye así a la línea de investigación del laboratorio de sistemas inteligentes de la Universidad Carlos III de Madrid y se suma esta funcionalidad a las ya existentes en la aplicación creada para el desarrollo este tipo de proyectos.

6.2. Desarrollos futuros

Expondremos ahora los posibles desarrollos futuros que quedan abiertos en la aplicación implementada tras el desarrollo del proyecto.

La funcionalidad implementada funciona bien con una iluminación correcta, mientras que con una iluminación pobre aumenta problemáticamente el error en la detección tanto facial como de los ojos. Queda como posible desarrollo futuro una mejora de este aspecto negativo.

Otro interesante desarrollo futuro, esta vez en el ámbito de la programación en Android, puede ser la internacionalización completa de la aplicación. Esto se puede realizar traduciendo los aspectos de la aplicación que no se han podido traducir en el desarrollo del proyecto, tales como los sonidos o los botones, ya que estos últimos provienen de imágenes prediseñadas. Además, siempre es positivo añadir más idiomas a la aplicación para hacerla más accesible al consumidor. En el ámbito de la programación en Android, otro aspecto que podría favorecer a la aplicación sería el desarrollo de la misma tanto para orientación vertical como horizontal (únicamente implementada para orientación horizontal en este proyecto).

Además, para añadir seguridad a la conducción, también sería interesante poder ampliar el número de síntomas detectables por la aplicación. Estos síntomas podrían ser bostezos, parpadeo continuo o desvío de la mirada de la carretera(Fig. 6.1).

Figura 6.1: Ejemplo de detección de la dirección de la mirada del conductor[4].

Otros posibles proyectos futuros de mayor envergadura podrían ser la implementación de una funcionalidad que aprovechara lo desarrollado en este proyecto para la detección de fatiga en el conductor mediante el dispositivo móvil e integrarlo en el automóvil en cuestión. Se podrían implementar algoritmos que permitieran la parada del automóvil en el caso de detectar fatiga en el conductor para mayor seguridad del mismo o desarrollar diferentes alarmas dentro del vehículo, como vibraciones del asiento del conductor o del volante.

Apéndice A

Código completo de la aplicación implementada

En este anexo se expondrá el código completo de la aplicación desarrollada, únicamente los archivos de la aplicación base del LSI que son de interés para este proyecto.

Empezaremos con el archivo "AndroidManifest.xml".

```
/**app/src/main/AndroidManifest.xml*/

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.lsi"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.
        WRITE_EXTERNAL_STORAGE" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature
        android:name="android.hardware.camera.autofocus"
        android:required="false" />
    <uses-feature
        android:name="android.hardware.camera.front"
        android:required="false" />
    <uses-feature
        android:name="android.hardware.camera.front.autofocus"
        android:required="false" />
    <uses-feature
        android:name="android.hardware.camera.flash"
        android:required="false" />

    <uses-sdk
```

```
27          android:minSdkVersion="11"
28          android:targetSdkVersion="19" />
29
30      <application
31          android:allowBackup="true"
32          android:icon="@drawable/ic_launcher"
33          android:label="@string/app_name"
34          android:theme="@style/AppTheme" >
35          <activity
36              android:name="com.lsi.MainActivity"
37              android:label="@string/app_name" >
38              <intent-filter>
39                  <action android:name="android.intent.action.MAIN" />
40
41                  <category android:name="android.intent.category.
                        LAUNCHER" />
42              </intent-filter>
43          </activity>
44          <activity
45              android:name="com.lsi.Ejemplos"
46              android:label="@string/title_activity_ejemplos" >
47          </activity>
48          <activity
49              android:name="com.lsi.ImageManipulationsActivity"
50              android:configChanges="keyboardHidden|orientation"
51              android:label="@string/
                    title_activity_image_manipulations"
52              android:screenOrientation="landscape" >
53          </activity>
54          <activity
55              android:name="com.lsi.QuienesSomos"
56              android:label="@string/title_activity_quienes_somos" >
57          </activity>
58          <activity
59              android:name="com.lsi.AppsLSI"
60              android:label="@string/title_activity_apps_lsi" >
61          </activity>
62          <activity
63              android:name="com.lsi.Puzzle15Activity"
64              android:configChanges="keyboardHidden|orientation"
65              android:label="@string/title_activity_puzzle15"
66              android:screenOrientation="landscape" >
67          </activity>
68          <activity
69              android:name="com.lsi.LibroActivity"
70              android:label="@string/title_activity_det_peatones"
71              android:screenOrientation="landscape" >
72          </activity>
73          <activity
74              android:name="org.opencv.samples.facedetect.FdActivity"
75              android:label="@string/title_activity_det_peatones" >
76          </activity>
77          <activity
```

```
78              android:name="com.lsi.DPActivity"
79              android:label="@string/title_activity_det_peatones"
80              android:screenOrientation="landscape" >
81          </activity>
82          <activity
83              android:name="com.lsi.FdActivity"
84              android:label="@string/title_activity_pd"
85              android:screenOrientation="landscape" >
86          </activity>
87          <activity
88              android:name="com.lsi.PedestrianActivity"
89              android:label="@string/title_activity_pd"
90              android:screenOrientation="landscape" >
91          </activity>
92          <activity
93              android:name="com.lsi.MainActDetPea"
94              android:label="@string/title_activity_main_act_det_pea"
                    >
95          </activity>
96          <activity
97              android:name="com.lsi.QSDetPea"
98              android:label="@string/title_activity_qsdet_pea" >
99          </activity>
100         <activity
101             android:name="com.lsi.CarBackActivity"
102             android:label="@string/title_activity_car_back"
103             android:screenOrientation="landscape" >
104         </activity>
105         <activity
106             android:name="com.lsi.MainActDetCarBack"
107             android:label="@string/
                    title_activity_main_act_det_car_back" >
108         </activity>
109         <activity
110             android:name="com.lsi.MainActDetFat"
111             android:label="@string/inicioDetFat" >
112         </activity>
113         <activity
114             android:name="com.lsi.QSDetFat"
115             android:label="@string/inicioDetFat" >
116         </activity>
117         <activity
118             android:name="com.lsi.DetFatigaActivity"
119             android:label="@string/inicioDetFat"
120             android:screenOrientation="landscape" >
121         </activity>
122     </application>
123
124 </manifest>
```

Seguiremos con el archivo “MainActivity.java”.

```
1  /**app/src/main/java/com.lsi/MainActivity.java*/
2
3  package com.lsi;
4
5  import android.net.Uri;
6  import android.os.Bundle;
7  import android.app.Activity;
8  import android.content.Intent;
9  import android.view.Menu;
10 import android.view.View;
11 import android.widget.ImageButton;
12
13 public class MainActivity extends Activity {
14
15    ImageButton botonInicio;
16    ImageButton botonLSI;
17    ImageButton botonEjemplos;
18    ImageButton botonQS;
19    ImageButton botonUC3M;
20
21    @Override
22    protected void onCreate(Bundle savedInstanceState) {
23       super.onCreate(savedInstanceState);
24       setContentView(R.layout.activity_main);
25
26       botonLSI = (ImageButton) findViewById(R.id.imageButtonLSI);
27       botonInicio = (ImageButton) findViewById(R.id.startButton);
28       botonEjemplos = (ImageButton) findViewById(R.id.ejemplosButton
             );
29       botonQS = (ImageButton) findViewById(R.id.whoWeAreButton);
30       botonUC3M = (ImageButton) findViewById(R.id.imageButtonUC3M);
31
32    }
33
34    @Override
35    public boolean onCreateOptionsMenu(Menu menu) {
36       // Inflate the menu; this adds items to the action bar if it
             is present.
37       getMenuInflater().inflate(R.menu.main, menu);
38       return true;
39    }
40
41    public void button_click(View v) {
42
43       int id = v.getId();
44       if (id == (R.id.imageButtonLSI)) {
45          goToUrl("http://portal.uc3m.es/portal/page/portal/
                dpto_ing_sistemas_automatica");
46       } else if (id == (R.id.startButton)) {
47          Intent i = new Intent(MainActivity.this, AppsLSI.class);
48          startActivity(i);
49       } else if (id == (R.id.ejemplosButton)) {
```

```
50            Intent ejemplos = new Intent(MainActivity.this, Ejemplos.
                  class);
51            startActivity(ejemplos);
52         } else if (id == (R.id.whoWeAreButton)) {
53            Intent qSomos = new Intent(MainActivity.this, QuienesSomos.
                  class);
54            startActivity(qSomos);
55         } else if (id == (R.id.imageButtonUC3M)) {
56            goToUrl("http://portal.uc3m.es");
57         }
58
59      }
60
61      public void goToUrl(String url) {
62
63         Uri uriUrl = Uri.parse(url);
64         Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uriUrl);
65         startActivity(launchBrowser);
66
67      }
68
69   }
```

Proseguimos con el contenido de los xml correspondiente al layout y al menú de esta clase.

```
1   /**app/src/main/res/layout/activity_main.xml*/
2
3   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
        android"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@drawable/fondo"
8       android:paddingBottom="@dimen/activity_vertical_margin"
9       android:paddingLeft="@dimen/activity_horizontal_margin"
10      android:paddingRight="@dimen/activity_horizontal_margin"
11      android:paddingTop="@dimen/activity_vertical_margin"
12      tools:context=".MainActivity" >
13
14      <LinearLayout
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17          android:layout_alignParentBottom="true"
18          android:layout_alignParentRight="true"
19          android:layout_alignParentTop="true"
20          android:background="@android:color/transparent"
21          android:gravity="center_horizontal"
22          android:orientation="vertical" >
23
24          <RelativeLayout
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
```

```
27              android:layout_weight="1" >
28
29              <ImageButton
30                  android:id="@+id/imageButtonLSI"
31                  android:layout_width="wrap_content"
32                  android:layout_height="wrap_content"
33                  android:background="@android:color/transparent"
34                  android:contentDescription="@string/enlaceWeb"
35                  android:focusable="true"
36                  android:focusableInTouchMode="false"
37                  android:onClick="button_click"
38                  android:src="@drawable/bot_lsi" />
39          </RelativeLayout>
40
41          <RelativeLayout
42              android:layout_width="match_parent"
43              android:layout_height="wrap_content"
44              android:layout_weight="1" >
45
46              <ImageButton
47                  android:id="@+id/startButton"
48                  android:layout_width="wrap_content"
49                  android:layout_height="wrap_content"
50                  android:layout_centerInParent="true"
51                  android:background="@android:color/transparent"
52                  android:contentDescription="@string/startApp"
53                  android:focusable="true"
54                  android:focusableInTouchMode="false"
55                  android:onClick="button_click"
56                  android:src="@drawable/bot_ap"
57                  android:layout_alignParentEnd="false" />
58          </RelativeLayout>
59
60          <RelativeLayout
61              android:layout_width="match_parent"
62              android:layout_height="wrap_content"
63              android:layout_weight="1" >
64
65              <ImageButton
66                  android:id="@+id/ejemplosButton"
67                  android:layout_width="wrap_content"
68                  android:layout_height="wrap_content"
69                  android:layout_centerInParent="true"
70                  android:background="@android:color/transparent"
71                  android:contentDescription="@string/ejemplos"
72                  android:focusable="false"
73                  android:focusableInTouchMode="false"
74                  android:gravity="center_vertical"
75                  android:onClick="button_click"
76                  android:src="@drawable/bot_ejemplos" />
77          </RelativeLayout>
78
79          <RelativeLayout
```

```
80              android:layout_width="match_parent"
81              android:layout_height="wrap_content"
82              android:layout_weight="1" >
83
84              <ImageButton
85                  android:id="@+id/whoWeAreButton"
86                  android:layout_width="wrap_content"
87                  android:layout_height="wrap_content"
88                  android:layout_centerInParent="true"
89                  android:background="@android:color/transparent"
90                  android:contentDescription="@string/whoWeAre"
91                  android:focusable="false"
92                  android:focusableInTouchMode="false"
93                  android:gravity="center_vertical"
94                  android:onClick="button_click"
95                  android:src="@drawable/bot_qs" />
96          </RelativeLayout>
97
98          <RelativeLayout
99              android:layout_width="match_parent"
100             android:layout_height="wrap_content"
101             android:layout_weight="1"
102             android:gravity="center_vertical"
103             android:padding="@dimen/activity_vertical_margin" >
104
105             <ImageButton
106                 android:id="@+id/imageButtonUC3M"
107                 android:layout_width="wrap_content"
108                 android:layout_height="wrap_content"
109                 android:layout_centerInParent="true"
110                 android:background="@android:color/transparent"
111                 android:contentDescription="@string/enlaceWeb"
112                 android:cropToPadding="true"
113                 android:focusable="false"
114                 android:focusableInTouchMode="false"
115                 android:gravity="center_vertical"
116                 android:onClick="button_click"
117                 android:scrollbarAlwaysDrawVerticalTrack="true"
118                 android:src="@drawable/bot_uc3m" />
119         </RelativeLayout>
120     </LinearLayout>
121
122 </RelativeLayout>
```

```
1  /**app/src/main/res/menu/main.xml*/
2
3  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
4
5      <item
6          android:id="@+id/action_settings"
7          android:orderInCategory="100"
8          android:showAsAction="never"
```

```
9           android:title="@string/action_settings"/>
10
11  </menu>
```

Ahora pasaremos a exponer el código correspondiente a la actividad "AppsLSI", tanto su archivo java como sus archivos xml de menú y layout.

```
1   /**app/src/main/java/com.lsi/AppsLSI.java*/
2
3   package com.lsi;
4
5   import android.app.Activity;
6   import android.app.AlertDialog;
7   import android.app.AlertDialog.Builder;
8   import android.content.DialogInterface;
9   import android.content.DialogInterface.OnClickListener;
10  import android.content.Intent;
11  import android.os.Bundle;
12  import android.view.Menu;
13  import android.view.View;
14
15  public class AppsLSI extends Activity {
16
17     String lista1,lista2, lista3, lista4, lista5;
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21        super.onCreate(savedInstanceState);
22        setContentView(R.layout.activity_apps_lsi);
23        lista1=getString(R.string.lista1);
24        lista2=getString(R.string.lista2);
25        lista3=getString(R.string.lista3);
26        lista4=getString(R.string.lista4);
27        lista5=getString(R.string.lista5);
28
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33        // Inflate the menu; this adds items to the action bar if it
              is present.
34        getMenuInflater().inflate(R.menu.apps_lsi, menu);
35        return true;
36     }
37
38     public void button_ej_click(View v) {
39        AlertDialog.Builder b = new Builder(this);
40        b.setTitle(lista1);
41        String[] types = {lista2,lista3,lista4,lista5};
42        b.setItems(types, new OnClickListener() {
43
44           @Override
45           public void onClick(DialogInterface dialog, int which) {
```

```
46
47              dialog.dismiss();
48              switch (which) {
49              case 0:
50                 Intent myIntent = new Intent(AppsLSI.this,
51                       MainActDetPea.class);
52                 startActivity(myIntent);
53                 break;
54              case 1:
55                 Intent myIntent2 = new Intent(AppsLSI.this,
56                       MainActDetCarBack.class);
57                 startActivity(myIntent2);
58                 break;
59              case 2:
60                 Intent myIntent3 = new Intent(AppsLSI.this,
61                       LibroActivity.class);
62                 startActivity(myIntent3);
63                 break;
64              case 3:
65                 Intent myIntent4 = new Intent(AppsLSI.this,
66                       MainActDetFat.class);
67                 startActivity(myIntent4);}
68
69           }
70
71        });
72
73        b.show();
74     }
75
76     public void button_click(View v) {
77        switch (v.getId()) {
78        case R.id.volver:
79           Intent i = new Intent(AppsLSI.this, MainActivity.class);
80           startActivity(i);
81           break;
82
83        }
84     }
85
86  }
```

```
1  /**app/src/main/res/layout/activity_apps_lsi.xml*/
2
3  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
       android"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@drawable/fondo"
8      android:paddingBottom="@dimen/activity_vertical_margin"
9      android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
10      android:paddingRight="@dimen/activity_horizontal_margin"
11      android:paddingTop="@dimen/activity_vertical_margin"
12      tools:context=".MainActivity" >
13
14      <LinearLayout
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17          android:layout_alignParentBottom="true"
18          android:layout_alignParentRight="true"
19          android:layout_alignParentTop="true"
20          android:background="@android:color/transparent"
21          android:gravity="center_horizontal"
22          android:orientation="vertical" >
23
24          <RelativeLayout
25              android:layout_width="match_parent"
26              android:layout_height="wrap_content"
27              android:layout_weight="1" >
28
29              <ImageButton
30                  android:id="@+id/btApps"
31                  android:layout_width="wrap_content"
32                  android:layout_height="wrap_content"
33                  android:layout_centerInParent="true"
34                  android:background="@android:color/transparent"
35                  android:contentDescription="@string/whoWeAre"
36                  android:focusable="false"
37                  android:focusableInTouchMode="false"
38                  android:gravity="center_vertical"
39                  android:onClick="button_ej_click"
40                  android:src="@drawable/bot_ea" />
41          </RelativeLayout>
42
43          <RelativeLayout
44              android:layout_width="match_parent"
45              android:layout_height="wrap_content"
46              android:layout_weight="1"
47              android:gravity="center_vertical"
48              android:padding="@dimen/activity_vertical_margin" >
49
50              <ImageButton
51                  android:id="@+id/volver"
52                  android:layout_width="wrap_content"
53                  android:layout_height="wrap_content"
54                  android:layout_centerInParent="true"
55                  android:background="@android:color/transparent"
56                  android:contentDescription="@string/whoWeAre"
57                  android:focusable="false"
58                  android:focusableInTouchMode="false"
59                  android:gravity="center_vertical"
60                  android:onClick="button_click"
61                  android:src="@drawable/bot_volver" />
62          </RelativeLayout>
```

```
63      </LinearLayout>
64
65  </RelativeLayout>
```

```
1   /**app/src/main/res/menu/apps_lsi.xml*/
2
3   <menu xmlns:android="http://schemas.android.com/apk/res/android" >
4
5       <item
6           android:id="@+id/action_settings"
7           android:orderInCategory="100"
8           android:showAsAction="never"
9           android:title="@string/action_settings"/>
10
11  </menu>
```

Mientras que sobre los anteriores archivos el único trabajo realizado sobre ellos fue modificar alguna de sus partes, pasamos ahora a exponer los que fueron creados en su totalidad para este proyecto, comenzando por los correspondientes a la clase "MainActDetFat", tanto su archivo java como el xml del layout.

```
1   /**app/src/main/java/com.lsi/MainActDetFat.java*/
2
3   package com.lsi;
4
5   import android.os.Bundle;
6   import android.app.Activity;
7   import android.content.Intent;
8   import android.view.View;
9
10  public class MainActDetFat extends Activity {
11
12      @Override
13      //Ejecutado al iniciar la actividad
14      protected void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);    /**Método que carga los
                parámetros guardados de la
16                                      anterior ejecución (
                                           savedInstanceState)*/
17          setContentView(R.layout.activity_main_act_det_fat);      /**
                Mostramos por pantalla el layout
18                                                      correspondiente*/
19      }
20
21      public void button_click(View v) {
22
23          switch (v.getId()) {    /**Obtenemos el Id del objeto sobre el
                que se ha pulsado
24                              e iniciamos en cada caso su actividad
```

```
                                correspondiente*/
25          case R.id.startButton:
26             Intent intent = new Intent(MainActDetFat.this,
27                   DetFatigaActivity.class);
28             startActivity(intent);
29             break;
30          case R.id.whoWeAreButton:
31             Intent intent2 = new Intent(MainActDetFat.this,
32                   QSDetFat.class);
33             startActivity(intent2);
34             break;
35          case R.id.volver:
36             Intent intent3 = new Intent(MainActDetFat.this,
37                   AppsLSI.class);
38             startActivity(intent3);
39             break;
40
41          default:
42             break;
43          }
44
45       }
46
47    }
```

```
1   /**app/src/main/res/layout/activity_main_act_det_fat.xml*/
2
3   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
        android"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@drawable/fondo"
8       android:paddingBottom="@dimen/activity_vertical_margin"
9       android:paddingLeft="@dimen/activity_horizontal_margin"
10      android:paddingRight="@dimen/activity_horizontal_margin"
11      android:paddingTop="@dimen/activity_vertical_margin"
12      tools:context=".MainActivity" >
13
14      <LinearLayout
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17          android:layout_alignParentBottom="true"
18          android:layout_alignParentRight="true"
19          android:layout_alignParentTop="true"
20          android:background="@android:color/transparent"
21          android:gravity="center_horizontal"
22          android:orientation="vertical"
23          android:paddingTop="10dp" >
24
25          <TextView
26              android:id="@+id/textView1"
```

```
27                android:layout_width="wrap_content"
28                android:layout_height="wrap_content"
29                android:layout_weight="1"
30                android:background="@android:color/transparent"
31                android:paddingTop="5dp"
32                android:text="@string/title_activity_main_act_det_fat"
33                android:textColor="@android:color/white"
34                android:textColorLink="@android:color/white"
35                android:textSize="18sp"
36                android:textStyle="italic"
37                android:typeface="serif"
38                android:gravity="center_vertical|center_horizontal" />
39
40            <RelativeLayout
41                android:layout_width="match_parent"
42                android:layout_height="wrap_content"
43                android:layout_weight="1" >
44
45                <ImageButton
46                    android:id="@+id/startButton"
47                    android:layout_width="wrap_content"
48                    android:layout_height="wrap_content"
49                    android:layout_centerInParent="true"
50                    android:background="@android:color/transparent"
51                    android:contentDescription="@string/whoWeAre"
52                    android:focusable="false"
53                    android:focusableInTouchMode="false"
54                    android:gravity="center_vertical"
55                    android:onClick="button_click"
56                    android:src="@drawable/bot_start" />
57            </RelativeLayout>
58
59            <RelativeLayout
60                android:layout_width="match_parent"
61                android:layout_height="wrap_content"
62                android:layout_weight="1" >
63
64                <ImageButton
65                    android:id="@+id/whoWeAreButton"
66                    android:layout_width="wrap_content"
67                    android:layout_height="wrap_content"
68                    android:layout_centerInParent="true"
69                    android:background="@android:color/transparent"
70                    android:contentDescription="@string/whoWeAre"
71                    android:focusable="false"
72                    android:focusableInTouchMode="false"
73                    android:gravity="center_vertical"
74                    android:onClick="button_click"
75                    android:src="@drawable/bot_qs" />
76
77            </RelativeLayout>
78
79            <RelativeLayout
```

```
80              android:layout_width="match_parent"
81              android:layout_height="wrap_content"
82              android:layout_weight="1" >
83
84              <ImageButton
85                  android:id="@+id/volver"
86                  android:layout_width="wrap_content"
87                  android:layout_height="wrap_content"
88                  android:layout_centerInParent="true"
89                  android:background="@android:color/transparent"
90                  android:contentDescription="@string/whoWeAre"
91                  android:focusable="false"
92                  android:focusableInTouchMode="false"
93                  android:gravity="center_vertical"
94                  android:onClick="button_click"
95                  android:src="@drawable/bot_volver" />
96          </RelativeLayout>
97      </LinearLayout>
98
99  </RelativeLayout>
```

Además, en nuestra aplicación también nos encontramos con el la clase "QSDetFat", cuyo layout y archivo java se expondrá a continuación.

```
1   /**app/src/main/java/com.lsi/QSDetFat.java*/
2
3   package com.lsi;
4
5   import android.app.Activity;
6   import android.os.Bundle;
7
8   public class QSDetFat extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_qs_det_fat);
14     }
15
16
17  }
```

```
1   /**app/src/main/res/layout/activity_qs_det_fat.xml*/
2
3   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
        android"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@drawable/fondo"
8       android:paddingBottom="@dimen/activity_vertical_margin"
```

```
9       android:paddingLeft="@dimen/activity_horizontal_margin"
10      android:paddingRight="@dimen/activity_horizontal_margin"
11      android:paddingTop="@dimen/activity_vertical_margin"
12      tools:context=".WhoWeActivity" >
13
14      <LinearLayout
15          android:layout_width="fill_parent"
16          android:layout_height="wrap_content"
17          android:layout_alignParentBottom="true"
18          android:layout_alignParentRight="true"
19          android:layout_alignParentTop="true"
20          android:background="@android:color/transparent"
21          android:gravity="center_horizontal"
22          android:orientation="vertical" >
23
24          <TextView
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
27              android:text="@string/desarrollo"
28              android:textColor="@android:color/white"
29              android:textSize="40sp"
30              android:textStyle="bold"
31              android:typeface="sans" />
32
33          <FrameLayout
34              android:layout_width="match_parent"
35              android:layout_height="wrap_content" >
36
37              <TextView
38                  android:id="@+id/textView1"
39                  android:layout_width="wrap_content"
40                  android:layout_height="wrap_content"
41                  android:layout_gravity="center_horizontal"
42                  android:background="@android:color/transparent"
43                  android:paddingTop="5dp"
44                  android:text="@string/descripfatiga"
45                  android:textColor="@android:color/white"
46                  android:textColorLink="@android:color/white"
47                  android:textSize="18sp"
48                  android:textStyle="italic"
49                  android:typeface="serif"
50                  android:gravity="center_horizontal" />
51
52          </FrameLayout>
53
54          <LinearLayout
55              android:layout_width="match_parent"
56              android:layout_height="wrap_content"
57              android:orientation="vertical"
58              android:paddingTop="5dp" >
59
60              <TextView
61                  android:id="@+id/textView2"
```

```
62                  android:layout_width="wrap_content"
63                  android:layout_height="wrap_content"
64                  android:layout_gravity="center_horizontal"
65                  android:paddingTop="3dp"
66                  android:text="@string/Sergio"
67                  android:textColor="@android:color/white"
68                  android:textSize="20sp"
69                  android:textStyle="italic|bold"
70                  android:typeface="serif" />
71
72              <TextView
73                  android:id="@+id/TextView05"
74                  android:layout_width="wrap_content"
75                  android:layout_height="wrap_content"
76                  android:layout_gravity="center_horizontal"
77                  android:paddingTop="2dp"
78                  android:text="@string/mail"
79                  android:textColor="@android:color/white"
80                  android:textSize="18sp"
81                  android:textStyle="italic" />
82          </LinearLayout>
83
84          <LinearLayout
85              android:layout_width="match_parent"
86              android:layout_height="wrap_content"
87              android:orientation="vertical"
88              android:paddingTop="5dp">
89
90              <TextView
91                  android:id="@+id/textView3"
92                  android:layout_width="wrap_content"
93                  android:layout_height="wrap_content"
94                  android:layout_gravity="center_horizontal"
95                  android:text="@string/tutor"
96                  android:textColor="@android:color/white"
97                  android:textSize="23sp"
98                  android:textStyle="bold" />
99
100             <TextView
101                 android:id="@+id/textView4"
102                 android:layout_width="wrap_content"
103                 android:layout_height="wrap_content"
104                 android:layout_gravity="center_horizontal"
105                 android:paddingTop="3dp"
106                 android:text="@string/tutorName"
107                 android:textColor="@android:color/white"
108                 android:textSize="20sp"
109                 android:textStyle="italic|bold" />
110
111             <TextView
112                 android:id="@+id/textView7"
113                 android:layout_width="wrap_content"
114                 android:layout_height="wrap_content"
```

```
                android:layout_gravity="center_horizontal"
                android:paddingTop="3dp"
                android:text="@string/tutorMail"
                android:textColor="@android:color/white"
                android:textSize="18sp"
                android:textStyle="italic"
                android:typeface="normal" />

        </LinearLayout>

    </LinearLayout>

</RelativeLayout>
```

Por último expondremos la clase en la cual es implementa la mayor parte de la funcionalidad desarrollada, “DetFatigaActivity”, con su respectivos archivos java y xml, tanto para el menú como para el layout.

```
/**app/src/main/java/com.lsi/DetFatigaActivity.java*/

/** APLICACION PARA LA DETECCIÓN DE LA FATIGA EN EL CONDUCTOR
    Desarrollada por Sergio Romero Salas
Tutor: Fernando García Fernandez */

package com.lsi;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;


import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2
    ;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

import android.app.Activity;
import android.content.Context;
import android.media.MediaPlayer;
```

```
33  import android.os.Bundle;
34  import android.os.CountDownTimer;
35  import android.util.Log;
36  import android.view.Menu;
37  import android.view.MenuItem;
38  import android.view.WindowManager;
39
40
41
42  public  class DetFatigaActivity extends Activity implements
43        CvCameraViewListener2 {
44
45     //DECLARACION DE VARIABLES
46
47     //Strings para el programa
48     String abralosojos ,conductornodetectado, ojosabiertos,
           pareydescanse;
49
50
51     //Etiqueta para logcat
52     public static final String TAG = "DETECTOR DE FATIGA ACTIVITY";
53
54     //Clasificadores para detección de ojos y cara
55     private CascadeClassifier eyescascade, facecascade;
56
57     // Colores
58     private static final Scalar FACE_RECT_COLOR = new Scalar(0, 0,
           255);
59     private static final Scalar AREA_RECT_COLOR = new Scalar(0, 255,
           0);
60     private static final Scalar EYE_CIRCLE_COLOR = new Scalar(255, 0,
            0);
61
62     //Sonidos
63     private MediaPlayer mire_carretera, pare_descanse;
64
65     //Variable para la carga de los xml de los clasificadores
66     private File mCascadeFile;
67
68     //Mínimo tamaño de cara relativo y absoluto, inicializado en 30%
69     private float mRelativeFaceSize = 0.3f;
70     private int mAbsoluteFaceSize = 0;
71
72     //Variables donde guardar los resultados encontrados
73     private MatOfRect eyes= new MatOfRect();
74     private MatOfRect face= new MatOfRect();
75
76     //Variables para la lógica de la alarma
77     private static int estado=0;
78     private static int eventos=0;
79     private static int cuentadetecciones=0;
80     private static int cuentaacabada=0;
81     private static int tempiniciado=0;
```

```
82
83      //Definición del contador
84      CountDownTimer T= new CountDownTimer(3000,1000){
85         @Override
86         public void onTick(long l) {
87         }
88         @Override
89         public void onFinish() {
90            cuentaacabada=1;
91         }
92      };
93
94      //Imagen en gris y en rgb
95      private Mat gray_img, rgb_img;
96
97      //Variable para cambio de cámara. 1 para frontal, 0 para trasera
            en nuestro dispositivo, en el
98      //caso de un dispositivo con otra disposición, la única
            diferencia sera que el primer cambio de
99      // cámara no se realizará, pero los posteriores sí.
100     private int mCameraId=1;
101
102     //Clase necesaria para acceder al uso de la cámara
103     private CameraBridgeViewBase mOpenCvCameraView;
104
105     //Clase para cargar la librería openCV. Ver
106     // http://docs.opencv.org/android/service/doc/BaseLoaderCallback.
            html
107     private BaseLoaderCallback mLoaderCallback = new
            BaseLoaderCallback(this) {
108        @Override
109        public void onManagerConnected(int status) {
110           switch (status) {
111              case LoaderCallbackInterface.SUCCESS: {
112
113                 Log.i(TAG, "OpenCV loaded successfully");
114
115              // Cargamos de los recrusos (raw) el clasificador para
                    la detección de ojos
116              // con el xml haarcascade_eye previamente entrenado que
                    nos ofrece OpenCV
117
118                 try {
119                    InputStream is = getResources().openRawResource(
120                          R.raw.haarcascade_eye);
121                    File cascadeDir = getDir("cascade", Context.
                          MODE_PRIVATE);
122                    mCascadeFile = new File(cascadeDir,
123                          "haarcascade_eye.xml");
124                    FileOutputStream os = new FileOutputStream(
                          mCascadeFile);
125
126                    byte[] buffer = new byte[4096];
```

```
127                     int bytesRead;
128                     while ((bytesRead = is.read(buffer)) != -1) {
129                        os.write(buffer, 0, bytesRead);
130                     }
131                     is.close();
132                     os.close();
133
134                     eyescascade = new CascadeClassifier(
135                           mCascadeFile.getAbsolutePath());
136                     if (eyescascade.empty()) {
137                        Log.e(TAG, "Failed to load cascade classifier")
                              ;
138                        eyescascade = null;
139                     } else
140                        Log.i(TAG, "Loaded cascade classifier from "
141                              + mCascadeFile.getAbsolutePath());
142
143                     cascadeDir.delete();
144
145                  } catch (IOException e) {
146                     e.printStackTrace();
147                     Log.e(TAG, "Failed to load cascade. Exception
                           thrown: " + e);
148                  }
149
150                  // Cargamos de los recrusos (raw) el clasificador
                        para la detección de cara
151                  // con el xml lbpcascade_frontalface previamente
                        entrenado???
152
153                  try {
154                     InputStream is = getResources().openRawResource(
155                           R.raw.lbpcascade_frontalface);
156                     File cascadeDir = getDir("cascade", Context.
                           MODE_PRIVATE);
157                     mCascadeFile = new File(cascadeDir,
158                           "lbpcascade_frontalface");
159                     FileOutputStream os = new FileOutputStream(
                           mCascadeFile);
160
161                     byte[] buffer = new byte[4096];
162                     int bytesRead;
163                     while ((bytesRead = is.read(buffer)) != -1) {
164                        os.write(buffer, 0, bytesRead);
165                     }
166                     is.close();
167                     os.close();
168
169                     facecascade = new CascadeClassifier(
170                           mCascadeFile.getAbsolutePath());
171                     if (facecascade.empty()) {
172                        Log.e(TAG, "Failed to load cascade classifier")
                              ;
```

```
173                         facecascade = null;
174                     } else
175                         Log.i(TAG, "Loaded cascade classifier from "
176                                 + mCascadeFile.getAbsolutePath());
177 
178                     cascadeDir.delete();
179 
180                 } catch (IOException e) {
181                     e.printStackTrace();
182                     Log.e(TAG, "Failed to load cascade. Exception
                            thrown: " + e);
183                 }
184                 //Habilitamos la vista de la cámara
185                 mOpenCvCameraView.enableView();
186             }
187             break;
188             default: {
189                 super.onManagerConnected(status);
190             }
191             break;
192         }
193     }
194 };
195 
196 
197 //Código para solucionar un error al arrancar la actividad por
        primera vez
198 static{
199     if(!OpenCVLoader.initDebug()){
200         //Handle initialization error
201     }
202 }
203 
204 @Override
205 public void onCreate(Bundle savedInstanceState) {
206     Log.i(TAG, "called onCreate");
207     super.onCreate(savedInstanceState);
208 
209     //Inicializamos los sonidos
210     mire_carretera=MediaPlayer.create(this, R.raw.
            mire_a_la_carretera);
211     pare_descanse=MediaPlayer.create(this, R.raw.pare_y_descanse);
212 
213     //Inicializamos los strings
214     abralosojos =getString(R.string.abralosojos);
215     conductornodetectado =getString(R.string.conductornodetectado)
            ;
216     ojosabiertos =getString(R.string.ojosabiertos);
217     pareydescanse =getString(R.string.pareydescanse);
218 
219     // mantiene la pantalla encendida mientras se ejecuta la
            activity
220     getWindow().addFlags(WindowManager.LayoutParams.
```

```
            FLAG_KEEP_SCREEN_ON);
221
222         //Inicializamos el layout y la cámara
223         setContentView(R.layout.activity_fat);
224         mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.
                fat_activity_surface_view);
225         mOpenCvCameraView.setCvCameraViewListener(this);
226     }
227
228
229     @Override
230     public void onPause() {
231         if (mOpenCvCameraView != null) {
232             mOpenCvCameraView.disableView();
233         }
234         super.onPause();
235     }
236
237     @Override
238     public void onResume() {
239         super.onResume();
240         OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_3, this
                ,
241                 mLoaderCallback);
242     }
243
244     @Override
245     public void onDestroy() {
246         super.onDestroy();
247         if (mOpenCvCameraView != null) {
248             mOpenCvCameraView.disableView();
249         }
250     }
251
252     @Override
253     public void onCameraViewStarted(int width, int height) {
254         gray_img = new Mat();
255         rgb_img = new Mat();
256     }
257
258     @Override
259     public void onCameraViewStopped() {
260         gray_img.release();
261         rgb_img.release();
262     }
263
264
265
266     @Override
267     public Mat onCameraFrame(CvCameraViewFrame inputFrame) {
268
269         //Preprocesamiento
270         gray_img = inputFrame.gray();
```

```
271         rgb_img=inputFrame.rgba();
272         Imgproc.equalizeHist(gray_img, gray_img);
273 
274         //Obtener el mínimo tamaño absoluto
275         if (mAbsoluteFaceSize == 0) {
276            int height = gray_img.rows();
277            if (Math.round(height * mRelativeFaceSize) > 0) {
278               mAbsoluteFaceSize = Math.round(height *
                      mRelativeFaceSize);
279            }
280         }
281 
282         //Deteccion de caras
283         facecascade.detectMultiScale(gray_img, face, 1.1, 2, 0,
284               new Size(mAbsoluteFaceSize,mAbsoluteFaceSize), new Size
                      ());
285 
286 
287         //Conversión a Array
288         Rect[] faceArray =face.toArray();
289 
290         //Aviso de conductor no detectado
291         if(faceArray.length==0){
292            Core.putText(rgb_img, conductornodetectado,
293                  new Point(0,rgb_img.rows()/2), 1, 1.5,
                         EYE_CIRCLE_COLOR, 2);
294         }
295 
296 
297         for(int i=0;i<faceArray.length;i++) {
298            //Pintado de rectangulo en la cara
299            Core.rectangle(rgb_img, faceArray[i].tl(), faceArray[i].br
                   (), FACE_RECT_COLOR
300               , 1);
301 
302            //Puntos necesario para marcar el area de los ojos
303            Point P1 = new Point(faceArray[i].tl().x,
304                  (0.2 * faceArray[i].br().y + 0.8 * faceArray[i].tl().
                         y));
305            Point P2 = new Point(faceArray[i].br().x,
306                  (0.55 * faceArray[i].br().y + 0.45 * faceArray[i].tl
                         ().y));
307 
308            //Rectangulo de la región de interés
309            Core.rectangle(rgb_img, P1, P2, AREA_RECT_COLOR, 3);
310 
311            //Creacion de faceROI (Region of Interest)
312            Rect AreaOjos=new Rect(P1,P2);
313            Mat faceROI = gray_img.submat(AreaOjos);
314 
315            //Deteccion de ojos
316            eyescascade.detectMultiScale(faceROI, eyes, 1.1, 13, 0,
317                  new Size(), new Size());
```

```
318
319             //Conversión a Array
320             Rect[] eyesArray = eyes.toArray();
321
322             //Circulo de los ojos detectados dentro del area de los
                    ojos
323
324             for (int j = 0; j < eyesArray.length; j++) {
325
326                Point center = new Point((eyesArray[j].br().x +
                       eyesArray[j].tl().x) / 2
327                   + AreaOjos.x,(eyesArray[j].br().y + eyesArray[j].
                          tl().y) / 2
328                   + AreaOjos.y);
329                int radius = (int) (eyesArray[j].br().x - eyesArray[j].
                       tl().x) / 2;
330
331                Core.circle(rgb_img, center, radius, EYE_CIRCLE_COLOR,
                       2);
332             }
333             int Ojosabiertos=eyesArray.length;
334             Core.putText(rgb_img, ojosabiertos + Ojosabiertos, new
                    Point(faceArray[i].tl().x,
335                faceArray[i].br().y), 1, 1.5,EYE_CIRCLE_COLOR, 2);
336
337             //Lógica para alarmas
338
339             switch (estado){
340                case 0:
341                   if(Ojosabiertos<2){
342                      estado=1;
343                      break;
344                   }
345                case 1:
346                   if(Ojosabiertos==2){
347                      cuentadetecciones++;
348                      if(cuentaacabada==1&&cuentadetecciones==3){
349                         eventos++;
350                         if(eventos>2){
351                            pare_descanse.start();
352                         }
353                      }
354                      if(cuentadetecciones==3){
355                         estado=0;
356                         cuentadetecciones=0;
357                         tempiniciado=0;
358                         T.cancel();
359                         cuentaacabada=0;
360                      }
361                   }
362                   else{
363                      if(tempiniciado==0){
364                         T.start();
```

```
365                         tempiniciado=1;
366                      }
367                      cuentadetecciones=0;
368                      if(cuentaacabada==1){
369                         Core.putText(rgb_img, abralosojos, new Point
370                               (faceArray[i].tl().x,P2.y),1, 1.5,
                                     EYE_CIRCLE_COLOR, 2);
371                         mire_carretera.start();
372                      }
373                   }
374                   break;
375             }
376             if(pare_descanse.isPlaying()){
377                Core.putText(rgb_img, pareydescanse, new Point(faceArray
                       [i].tl().x,
378                         P2.y), 1, 1.5, EYE_CIRCLE_COLOR, 2);
379             }
380          }
381
382    /*******************************************************/
383          return rgb_img;
384       }
385
386
387
388       //FUNCIONES PARA LA CREACIÓN Y ADQUISICIÓN DE DATOS DEL MENÚ DE
              LA ACTIVIDAD
389
390       @Override
391       public boolean onCreateOptionsMenu(Menu menu) {
392          Log.i(TAG, "called onCreateOptionsMenu");
393          // Inflate the menu; this adds items to the action bar if it
                 is present.
394          getMenuInflater().inflate(R.menu.det_fatiga, menu);
395
396          return true;
397
398       }
399
400       @Override
401       public boolean onOptionsItemSelected(MenuItem item) {
402          Log.i(TAG, "called onOptionsItemSelected; item seleccionado: "
                    + item);
403          switch (item.getItemId()) {
404             case R.id.menu_50:
405                setMinFaceSize(0.5f);
406                return true;
407
408             case R.id.menu_40:
409                setMinFaceSize(0.4f);
410                return true;
411
412             case R.id.menu_30:
```

```
413                 setMinFaceSize(0.3f);
414                 return true;
415
416             case R.id.menu_20:
417                 setMinFaceSize(0.2f);
418                 return true;
419
420             case R.id.menu_cambiarcamara:
421                 mCameraId = mCameraId^1; //Operación lógica not para
                        cambiar entre 0 y 1
422                 mOpenCvCameraView.disableView(); //Deshabilitamos la
                        vista
423                 mOpenCvCameraView.setCameraIndex(mCameraId); //Cambiamos
                         la cámara
424                 mOpenCvCameraView.enableView();     //Habilitamos la
                        vista de nuevo
425                 return true;
426
427             default:
428                 return super.onOptionsItemSelected(item);
429         }
430     }
431
432     //Funcion para cambiar el tamaño minimo de la cara relativo
            llamada desde el menú.
433     //Ponemos el minimo absoluto a 0 para que se actualice en
            onCameraFrame.
434     private void setMinFaceSize(float faceSize) {
435         mRelativeFaceSize = faceSize;
436         mAbsoluteFaceSize = 0;
437     }
438
439 }
```

```
1  /**app/src/main/res/menu/det_fatiga.xml*/
2
3  <?xml version="1.0" encoding="utf-8"?>
4  <menu xmlns:android="http://schemas.android.com/apk/res/android">
5      <item
6          android:id="@+id/menu_cambiarcamara"
7          android:orderInCategory="0"
8          android:showAsAction="ifRoom|withText"
9          android:title="@string/cambiarcamara"/>
10
11     <item
12         android:id="@+id/menu_20"
13         android:orderInCategory="0"
14         android:showAsAction="never"
15         android:title="@string/fat20"/>
16     <item
17         android:id="@+id/menu_30"
18         android:orderInCategory="0"
```

```
19          android:showAsAction="never"
20          android:title="@string/fat30"/>
21      <item
22          android:id="@+id/menu_40"
23          android:orderInCategory="0"
24          android:showAsAction="never"
25          android:title="@string/fat40"/>
26      <item
27          android:id="@+id/menu_50"
28          android:orderInCategory="0"
29          android:showAsAction="never"
30          android:title="@string/fat50"/>
31
32  </menu>
```

```
1   /**app/src/main/res/layout/activity_fat.xml*/
2
3   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
        android"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       xmlns:opencv="http://schemas.android.com/apk/res-auto" >
8
9       <org.opencv.android.JavaCameraView
10          android:id="@+id/fat_activity_surface_view"
11          android:layout_width="match_parent"
12          android:layout_height="match_parent"
13          opencv:show_fps="true"
14          opencv:camera_id="front"
15          android:longClickable="false" />
16
17
18  </LinearLayout>
```

Además, también se ha de añadir los archivos de strings donde están contenidas las definiciones de los mismos, tanto en inglés como en el idioma por defecto(español).

```
1   /**app/src/main/res/values/strings.xml*/
2
3   <?xml version="1.0" encoding="utf-8"?>
4   <resources>
5
6       <string name="app_name" translatable="false">LSIappDet</string>
7       <string name="enlaceWeb">Enlace a la web del Dpto.</string>
8       <string name="action_settings">Configuracion</string>
9       <string name="startApp">Iniciar aplicación</string>
10      <string name="whoWeAre">¿Quiénes somos?</string>
11      <string name="title_activity_who_we">¿Quiénes somos?</string>
12      <string name="creador">CREADOR</string>
```

```
13      <string name="desarrollo">DESARROLLO</string>
14      <string name="hello_world">!Hola Mundo!</string>
15      <string name="miNombre" translatable="false">Alejandro Ramos
            López</string>
16      <string name="descripBase">Aplicación base para proyectos del
            LSI-UC3M desarrollada por:</string>
17      <string name="descripDetPea">Aplicación de detección de peatones
             desarrollada para el LSI-UC3M por:</string>
18      <string name="descripDetCoches">Aplicación de detección de
            peatones desarrollada para el LSI-UC3M por:</string>
19      <string name="inicioDetPea">APP PARA DETECCIÓN DE PEATONES</
            string>
20      <string name="title_activity_hello_open_cv" translatable="false"
            >HelloOpenCvActivity</string>
21      <string name="title_image_manipulations_surface" translatable="
            false">ImageManipulationActivity</string>
22      <string name="tutor">TUTOR</string>
23      <string name="tutorName" translatable="false">Fernando García
            Fernández</string>
24      <string name="tutorMail" translatable="false">fegarcia@ing.uc3m.
            es</string>
25      <string name="jdpname">DIRECTOR DE PROYECTO</string>
26      <string name="jefeProyecto" translatable="false">Juan Carmona
            Fernández</string>
27      <string name="direcProyectoMail" translatable="false">
            carmonauc3m@gmail.com</string>
28      <string name="contacto">CONTACTO</string>
29      <string name="correos">EMAILS</string>
30      <string name="correo1" translatable="false">alexrl1990@gmail.com
            </string>
31      <string name="correo2" translatable="false">alexrl1990@hotmail.
            com</string>
32      <string name="correo3" translatable="false">100080835@alumnos.
            uc3m.es</string>
33      <string name="ejemplos">Ejemplos de OpenCV</string>
34      <string name="title_activity_ejemplos">Ejemplos de OpenCV</
            string>
35      <string name="imageManipulationEj">Ejemplo de manipulación de
            imagen</string>
36      <string name="title_activity_tutorial_2_activity" translatable="
            false">Tutorial2Activity</string>
37      <string name="tutorial_3" translatable="false">Tutorial3Activity
            </string>
38      <string name="title_activity_image_manipulation_zoom"
            translatable="false">Ej Image Manipulation</string>
39      <string name="title_activity_image_manipulation" translatable="
            false">Manipulacion Imagen Zoom</string>
40      <string name="title_activity_quienes_somos">¿Quiénes somos?</
            string>
41      <string name="title_activity_image_manipulations" translatable="
            false">Ejemplo Image Manipulation</string>
42      <string name="title_activity_proyectos_lsi">Proyectos LSI</
            string>
```

```
43      <string name="title_activity_apps_lsi" translatable="false">
            AppsLSI</string>
44      <string name="menu_toggle_tile_numbers">Mostrar/ocultar nombres
            de los títuls</string>
45      <string name="menu_start_new_game">Empezar nuevo juego</string>
46      <string name="title_activity_puzzle15" translatable="false">
            Puzzle15Activity</string>
47      <string name="title_activity_det_peatones">Detector Peatones</
            string>
48      <string name="borrar">Borrar</string>
49      <string name="editar">Editar</string>
50      <string name="menu_next_camera">Siguiente Camara</string>
51      <string name="menu_take_photo">Hacer foto</string>
52      <string name="photo_delete_prompt_message">Foto guardada en
            Galería. ¿Seguro que desea borrarla?</string>
53      <string name="photo_delete_prompt_title">¿Borrar foto?</string>
54      <string name="photo_error_message">Error al guardar la imagen</
            string>
55      <string name="photo_edit_chooser_title">Editar foto con?</string
            >
56      <string name="photo_send_chooser_title">Compartir foto con?</
            string>
57      <string name="photo_send_extra_subject">Mi foto de la app DetPea
            </string>
58      <string name="photo_send_extra_text">Verificar mi foto de DetPea
            </string>
59      <string name="compartir">Compartir</string>
60      <string name="menu_next_curve_filter">Siguiente curva</string>
61      <string name="menu_next_mixer_filter">Siguiente mezclador</
            string>
62      <string name="menu_next_convolution_filter">Siguiente Kernel</
            string>
63      <string name="title_activity_pd" translatable="false">PDActivity
            </string>
64      <string name="tituloSpinner">Proyectos de LSI</string>
65      <string name="pedDet" translatable="false">PedestrianDetector</
            string>
66      <string name="carBackDet" translatable="false">CarBackDetector</
            string>
67      <string name="inicioDetCoches">APP PARA DETECCIÓN DE COCHES</
            string>
68      <string name="deteccionfatiga">Deteccion Fatiga</string>
69
70      <string-array name="entriesSpinner">
71          <item>Selecciona una app</item>
72          <item>Detector de Peatones</item>
73          <item>Detector de Coches</item>
74      </string-array>
75
76      <string name="title_activity_main_act_det_pea">Detección de
            peatones</string>
77      <string name="title_activity_qsdet_pea" translatable="false">
            QSDetPea</string>
```

```
78      <string name="title_activity_car_back">Detección de Coches</
            string>
79      <string name="title_activity_main_act_det_car_back">Main
            Detector Coches</string>
80
81      <string name="title_activity_main_act_det_fat">APP PARA
            DETECCIÓN DE FATIGA</string>
82      <string name="inicioDetFat">Detección De Fatiga</string>
83      <string name="Sergio" translatable="false">Sergio Romero Salas</
            string>
84      <string name="mail" translatable="false">100291765@alumnos.uc3m.
            es</string>
85      <string name="descripfatiga">Aplicación de detección de fatiga
            desarrollada para el LSI-UC3M por:</string>
86      <string name="cambiarcamara">Cambiar Cámara</string>
87      <string name="fat50">Tamaño Rostro 50%</string>
88      <string name="fat40">Tamaño Rostro 40%</string>
89      <string name="fat30">Tamaño Rostro 30%</string>
90      <string name="fat20">Tamaño Rostro 20%</string>
91      <string name="conductornodetectado">ATENCION, CONDUCTOR NO
            DETECTADO</string>
92      <string name="ojosabiertos">Ojos abiertos: </string>
93      <string name="abralosojos">POR FAVOR, ABRA LOS OJOS</string>
94      <string name="pareydescanse">POR FAVOR, PARE Y DESCANSE</string>
95
96      <string name="lista1">Seleccione la app</string>
97      <string name="lista2">Detector de Peatones</string>
98      <string name="lista3">Detector de Coches</string>
99      <string name="lista4">Ejemplo Libro OpenCV</string>
100     <string name="lista5">Deteccion de fatiga</string>
101
102 </resources>
```

```
1   /**app/src/main/res/values-en/strings.xml*/
2   <?xml version="1.0" encoding="utf-8"?>
3   <resources>
4       <string name="whoWeAre">Who we are?</string>
5       <string name="tituloSpinner">LSI Projects</string>
6       <string name="title_activity_who_we">Who we are?</string>
7       <string name="title_activity_quienes_somos">Who we are?</string>
8       <string name="title_activity_proyectos_lsi">LSI Projects</string
            >
9       <string name="title_activity_main_act_det_pea">Pedestrian
            Detection</string>
10      <string name="title_activity_main_act_det_fat">FATIGUE DETECTION
             APP</string>
11      <string name="title_activity_main_act_det_car_back">Car
            Detection Main</string>
12      <string name="title_activity_ejemplos">OpenCV Examples</string>
13      <string name="title_activity_det_peatones">Pedestrian Detection
            </string>
14      <string name="startApp">Start App</string>
```

```
15      <string name="photo_send_extra_subject">My DetPea app photo</
            string>
16      <string name="photo_send_chooser_title">Share picture with?</
            string>
17      <string name="photo_error_message">Error saving image</string>
18      <string name="photo_edit_chooser_title">Edit picture with?</
            string>
19      <string name="photo_delete_prompt_title">Delete picture?</string
            >
20      <string name="photo_delete_prompt_message">Picture saved in
            galery. Are you sure you want to delete it?</string>
21      <string name="pareydescanse">PLEASE, STOP AND REST</string>
22      <string name="ojosabiertos">Opened eyes: </string>
23      <string name="menu_take_photo">Take picture</string>
24      <string name="jdpname">PROJECT DIRECTOR</string>
25      <string name="inicioDetPea">PEDESTRIAN DETECTION APP</string>
26      <string name="inicioDetFat">Fatigue Detection</string>
27      <string name="inicioDetCoches">CAR DETECTION APP</string>
28      <string name="imageManipulationEj">Image Manipulation Example</
            string>
29      <string name="enlaceWeb">Department Web Link</string>
30      <string name="ejemplos">OpenCV Examples</string>
31      <string name="editar">Edit</string>
32      <string name="deteccionfatiga">Fatigue Detection</string>
33      <string name="descripfatiga">Fatigue Detection App developed for
             LSI-UC3M by:</string>
34      <string name="descripDetPea">Pedestrian Detection App developed
            for LSI-UC3M by:</string>
35      <string name="descripDetCoches">Pedestrian Detection App
            developed for LSI-UC3M by:</string>
36      <string name="descripBase">Base App for projects of LSI-UC3M
            developed by:</string>
37      <string name="desarrollo">DEVELOPMENT</string>
38      <string name="creador">CREATOR</string>
39      <string name="correos">EMAILS</string>
40      <string name="contacto">CONTACT</string>
41      <string name="conductornodetectado">ATTENTION, DRIVER NOT
            DETECTED</string>
42      <string name="compartir">Share</string>
43      <string name="cambiarcamara">Change Cam</string>
44      <string name="borrar">Delete</string>
45      <string name="abralosojos">PLEASE, OPEN YOUR EYES</string>
46      <string name="action_settings">Settings</string>
47      <string name="fat30">Face Size 30%</string>
48      <string name="fat40">Face Size 40%</string>
49      <string name="fat50">Face Size 50%</string>
50      <string name="fat20">Face Size 20%</string>
51      <string name="hello_world">Hello World!</string>
52      <string name="menu_next_camera">Next Cam</string>
53      <string name="menu_next_convolution_filter">Next Kernel</string>
54      <string name="menu_next_curve_filter">Next Curve</string>
55      <string name="menu_next_mixer_filter">Next Mixer</string>
56      <string name="menu_start_new_game">Start new game</string>
```

```
57      <string name="menu_toggle_tile_numbers">Show/hide tile numbers</
            string>
58      <string name="photo_send_extra_text">Check out my photo from
            DetPea</string>
59      <string name="title_activity_car_back">Car Back Detector</string
            >
60      <string name="tutor">TUTOR</string>
61      <string name="lista1">Select the App</string>
62      <string name="lista2">Pedestrian Detector</string>
63      <string name="lista3">Car Detector</string>
64      <string name="lista4">OpenCv Book Example</string>
65      <string name="lista5">Fatigue Detector</string>
66  </resources>
```

Bibliografía

[1] Android Studio [Online]. Disponible en http://developer.android.com/intl/es/sdk/index.html.

[2] Carmelo Marín. Detector de ojos [Online]. Disponible en http://acodigo.blogspot.com.es/2013/08/detector-de-ojos.html.

[3] Circula Seguro. (2013, Diciembre) Sistemas para evitar el ángulo muerto de los espejos [Online]. Disponible en http://www.circulaseguro.com/sistemas-para-evitar-el-angulo-muerto/.

[4] Circula Seguro. (2014, Febrero) Sistemas de detección de la fatiga y falta de concentración al volante [Online]. Disponible en http://www.circulaseguro.com/sistemas-de-deteccion-de-la-fatiga-y-falta-de-concentracion-al-volante/.

[5] edX y UC3M. Â¿Hasta dónde pueden ver las máquinas? Curso de Visión por computador utilizando OpenCV.

[6] El País.(2013, Octubre) 15 meses, vida media de un móvil [Online]. Disponible en http://tecnologia.elpais.com/tecnologia/2013/10/25/actualidad/1382711542_793144.html.

[7] Google. Google Self-Driving Car Project [Online]. Disponible en http://www.google.com/selfdrivingcar/.

[8] Guevara, Marta Lucía; Echeverry, Julian David; Ardila Urueña, William. 2008. "Detección de rostros en imágenes digitales usando clasificadores en cascada". Scientia Et Technica, num. Junio, pp. 1-5.

[9] Hipertextual. (2015, Septiembre) Conductores kamikazes, Bosch quiere acabar con vosotros [Online]. Disponible en http://hipertextual.com/2015/09/conductores-kamikazes-bosch.

[10] JAVA. Learn About Java Technology [Online]. Disponible en https://www.java.com/en/about/.

[11] K. Hong. Object Detection : Face Detection using Haar Cascade Classfiers [Online]. Disponible en http://www.bogotobogo.com/python/opencv_python/python_opencv3_image_object_detection_face_detection_haar_cascade_classifiers.php.

[12] LSI UC3M. (2013, Mayo) Sitemas Inteligentes de Transporte [Online]. Disponible en http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes/sis_int_transporte/.

[13] LSI UC3M. (2015, Marzo) Laboratorio de Sistemas Inteligentes [Online]. Disponible en http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes.

[14] Motor Pasión. (2015, Marzo) El nuevo Ford S-MAX estrenará limitador inteligente de velocidad [Online]. Disponible en http://www.motorpasion.com/ford/el-nuevo-ford-s-max-estrenara-limitador-inteligente-de-velocidad.

[15] OMS. (2014, Mayo) Las 10 causas principales de defunción en el mundo [Online]. Disponible en http://www.who.int/mediacentre/factsheets/fs310/es/.

[16] OpenCV. Base Loader Callback Interface Implementation [Online]. Disponible en http://docs.opencv.org/android/service/doc/baseloadercallback.html.

[17] OpenCV. Cascade Classification [Online]. Disponible en http://docs.opencv.org/doc/tutorials/introduction/android_binary_package/dev_with_ocv_on_android.html#hello-opencv-sample.

[18] OpenCV. Cascade Classification [Online]. Disponible en http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html?highlight=cascadeclassifier#cv.haardetectobjects.

[19] OpenCV. Cascade Classifier [Online]. Disponible en http://docs.opencv.org/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html.

[20] OpenCV. Cascade Classifier Training [Online]. Disponible en http://docs.opencv.org/doc/user_guide/ug_traincascade.html.

[21] RTVE. (2015, Febrero) Los accidentes de tráfico descienden a la quinta posición como causa de muerte no natural [Online]. Disponible en http://www.rtve.es/noticias/20150227/accidente-trafico-dejan-ser-primera-causa-muerte-no-natural-pasan-quinto-lugar/1105860.shtml.

[22] SONY. Sony Xperia M [Online]. Disponible enhttp://www.sonymobile.com/es/products/phones/xperia-m/specifications#tabs.

[23] Stackoverflow - http://stackoverflow.com/.

[24] Volkswagen. Front Assist [Online]. Disponible en http://www.volkswagenferper.com/front-assist.

[25] Fred Birchmore. Using color-based features with boosting techniques to detect and recognize a surrogate AK-47. 2007.

[26] Gary Bradski and Adrian Kaehler. Learning OpenCV. 2008.

[27] Clifton Craig and Adam Gerber. Learn Android Studio: Build Android apps quickly and effectively. 2015.

[28] Natalia Morán Cruz. Trabajo fin de grado. desarrollo de un sistema avanzado de asistencia a la conducciión en tiempo real para la detecciión de peatones en entornos urbanos complejos. 2013.

[29] Alejandro Ramos López. Trabajo fin de grado. implementación de algoritmos de visión por computador en plataforma Android.